NORTHWESTERN UNIVERSITY

STRUCTURAL AND NETWORK-BASED METHODS FOR KNOWLEDGE-BASED SYSTEMS

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

field of Computer Science

By

ABHISHEK SHARMA

EVANSTON, ILLINOIS

DECEMBER 2011

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **DEC 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**
4. TITLE AND SUBTITLE **Structural and Network-based Methods for Knowledge-based Systems**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Northwestern University,2133 Sheridan Road,Evanston,IL,60208**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In recent years, there has been considerable interest in Learning by Reading and Machine Reading systems. These systems can learn thousands or even millions of facts from the Web. But to exploit this opportunity, we must address two issues: (a) Efficient first-order reasoning systems can be built today only for small-to-medium sized knowledge bases and by careful hand-tuning of inference mechanisms and representations. As knowledge bases grow, better ways to automatically use such knowledge efficiently must be found. (b) Secondly, how do reasoning systems that learn evolve over time? Characterizing the evolution of these systems is important for understanding their limitations and gaining insights into the interplay between learning and reasoning. In this work, we address these problems by focusing on the systemic properties of knowledge-based systems. We show that ideas from the fields of complex networks, SAT solving, and game theory can be used to improve Q/A performance in large knowledge-based learning systems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**142**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

ABSTRACT

Structural and Network-based Methods for Knowledge-based Systems

In recent years, there has been considerable interest in Learning by Reading and Machine Reading systems. These systems can learn thousands or even millions of facts from the Web. But to exploit this opportunity, we must address two issues: (a) Efficient first-order reasoning systems can be built today only for small-to-medium sized knowledge bases and by careful hand-tuning of inference mechanisms and representations. As knowledge bases grow, better ways to automatically use such knowledge efficiently must be found. (b) Secondly, how do reasoning systems that learn evolve over time? Characterizing the evolution of these systems is important for understanding their limitations and gaining insights into the interplay between learning and reasoning. In this work, we address these problems by focusing on the systemic properties of knowledge-based systems. We show that ideas from the fields of complex networks, SAT solving, and game theory can be used to improve Q/A performance in large knowledge-based learning systems.

Acknowledgements

I would like to thank Professor Kenneth D. Forbus for advising this research. This work would not be possible without his advice and encouragement.

I thank Chris Riesbeck, Doug Downey and Johan de Kleer for their advice and help.

I am also grateful to DARPA IPTO and Office of Naval Research for supporting this work.

I thank my family members for supporting my education and providing support and encouragement.

Dedicated to Mrs. Veena Sharma

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation and Contributions

Question Answering (Q/A) modules are needed for many applications. Since typical information retrieval techniques are inadequate for complex Q/A tasks, deductive reasoning plays an important role in many Artificial Intelligence (AI) systems. We believe that first-order deductive reasoning in large KBs will be needed for building true Q/A systems. In recent years, there has been considerable interest in building large Learning by Reading systems [Barker et al 2007; Forbus et al 2007, Mulkar et al 2007]. DARPA's Machine Reading program[1] and the European Union's Large Knowledge Collider project[2] are aimed at creating massive knowledge bases and reasoning with them. Improvement in Q/A technologies should help in improving the performance of these systems.

However, there are at least three problems which impede our progress towards practical knowledge-based AI systems:

- Efficient first-order reasoning systems can be built today only for small-to-medium sized knowledge bases and by careful hand-tuning of inference mechanisms and representations. There are two reasons to seek more general solutions. First, hand-tuning and careful crafting do not scale as the size of the knowledge base grows. For example, queries that fail in large knowledge bases frequently take hours to fail.[3] There is still no evidence that general-purpose first-order reasoning in such knowledge bases can regularly be performed in order of a few minutes per query in today's computing environments. The second problem is that knowledge bases are no longer being built entirely by hand. Advances in machine learning and knowledge capture provide the opportunity to automatically construct large knowledge bases [Etzioni et al 2005, Forbus et al 2007]. But to exploit this opportunity, we must be able to reason with them effectively. Therefore, we need to find ways to reduce the hardness of first-order reasoning systems.
- Secondly, knowledge base construction is difficult and tedious. Even the largest knowledge bases do not have sufficient axioms for attaining a reasonable level of Q/A performance. Therefore, current deductive reasoning systems have to deal with the problem of knowledge gaps and

[1] http://www.darpa.mil/IPTO/solicit/baa/BAA-09-03_PIP.pdf
[2] http://www.larkc.eu/
[3] cf. www.projecthalo.com/content/docs/

missing reasoning chains. Moreover, knowledge-based systems should not expect perfectly correct information. Ideally, a knowledge-base system should reduce its dependence on axioms by finding new methods for inferring answers. They should also be able to learn correct reasoning patterns with the help of proper user feedback.

- Finally, the study of large-scale knowledge based systems has mainly proceeded along the lines of measuring their efficacy in improving the amount of knowledge in the system. These are evolving systems: Over time, they learn new ground facts and new predicates and collections are introduced, thereby altering the structure of their knowledge base (KB). Given the nascent state of the art, so far the learned knowledge is typically small compared to the knowledge base the system starts with. Hence the size of the KB is constant for all practical purposes, and the set of axioms it uses for reasoning will be stable and continue to perform as they did before. But what will happen to reasoning performance as the state of the art improves, and the number of facts the system has learned by reading (or using machine reading techniques) dwarfs its initial endowment? There hasn't been any study of how the systemic properties of large learning systems would evolve.

Therefore, in this work, we propose some methods for solving these problems. In particular, this work makes following contributions:

- **Optimization of Horn-axioms for Efficient Reasoning:** In the first part of this work, we propose an axiom selection method for addressing the problem of inefficient reasoning. It is well-known that knowledge representation choices play a crucial role in determining the hardness of problems. Work in SAT problem solving is moving towards understanding the structure of problems and using it for the design of better heuristics [Kautz & Selman 2007, Walsh 1999]. First-order knowledge bases, though structured, are highly complex networks of concepts. Predicates are connected through different types of relationships, defining intricate networks. Understanding KB structure is fundamental to many important knowledge representation and reasoning problems. Therefore, we identify some systemic properties of knowledge bases which play an important role in determining the hardness of problems. We then describe an algorithm, which constructs an efficient set of Horn clauses from a first-order knowledge base consisting of general clauses. It takes as input a set of target queries, information about the current distribution of ground facts in the KB, and information about the kinds of facts that can be provided in the future (for example, via machine learning or knowledge capture or learning by reading). We then improve the performance of this algorithm by using some topological properties of predicate connectivity. We use the distribution of known facts and what might be added to identify difficult and less useful regions of the KB, and use that information to prune inefficient axioms, which improves performance. We also study the importance of maximum out-degree of nodes in the search graph for controlling the navigability of search spaces. We find that networks which have high degree nodes, called *hubs*, are inherently unsuitable for focused search. We study the family of search spaces, from disconnected to scale-free, by varying this parameter and show that it helps in identifying efficient sets of axioms.

- **Static Analysis Method for Improving Commonsense Reasoning:** In the second part of this work, we describe how reasoning engines can improve their performance by exploiting statistical properties of knowledge bases. These properties are calculated via an off-line static analysis, and provide estimates of the likelihood of success for particular axioms and search paths. We present three approaches. (1) We describe a method for analyzing axioms and ground facts to approximate the *parameterized deductive closure*, i.e. an estimate of kind of things which could be proved when inference is restricted to depth *d*. This database provides a model of system's knowledge that is used to prune queries which are not likely to be useful, thereby improving performance. (The pruned queries are also potentially useful in determining knowledge goals for learning.) (2) we show that, frequently, the antecedents of axioms jointly impose very strong constraints on the types of entities that can satisfy them, which can be used to select whether or not an axiom will be useful in a specific query. Although the axioms we use are quantified, first-order knowledge, we are inspired by constraint processing methods in the propositional CSP literature. We use similar ideas to develop the notion of the *global constraint space,* which is much smaller than constraints induced by predicates. Our experiments show that reasoning in this space leads to an order of magnitude improvement in time and space with no loss in completeness. (3) Finally, we show that proofs generated over a training set of queries can be used to estimate the likelihood of an axiom contributing to a successful proof. These likelihoods are used to prune less useful rules. Experiments show that such an approach leads to significant savings in time and memory.

- **Plausible Reasoning:** In the third part of the work, we concentrate on the problem of knowledge gaps in KBs. Since the set of axioms in incomplete even in largest KBs, we must find ways to reduce our dependence on them. We show how to integrate graph search, higher-order knowledge representation, and reinforcement learning to learn reliable patterns of plausible reasoning from ground facts. Given a fully grounded query, we show how to incrementally search the facts which mention the entities in it guided by a set of *plausible inference patterns* (PIPs). PIPs are similar to knowledge patterns [Clark *et al* 2000], but are expressed in terms of higher-order concepts in the knowledge base, specifically predicate type information. Since the number of predicate types is much smaller than the number of predicates, this greatly reduces the size of search space. We also believe that a representation in terms of predicate types is more natural and intuitive. We show that the quality of inference chains of PIPs can be learned by reinforcement learning. Experiments show that such an approach helps in improving Q/A performance.

- **Modeling the Evolution of Learning Systems**: The size of KBs virtually remains constant in current learning by reading systems because the number of facts learnt remains small compared to the size of the KB. (For example, learning 2,000-3,000 facts does not change the dynamics of reasoning when the initial KB contains 1.2 million ground facts.) We are interested in knowing how the systemic properties of KBs would change when the size of KBs increases significantly. To understand these issues, we introduce an *inverse ablation model*. The basic idea is to take the contents of a large knowledge base (here, ResearchCyc[4]) and make a simulation of the initial endowment of a learning by reading system by removing most of the facts. Reasoning performance is tested on this initial endowment, including the generation of learning goals. The operation of a learning component is simulated by gathering facts from the ablated portion of the KB that satisfy the learning goals, and adding those to the test KB. Performance is then tested again, new learning goals are generated, and the process continues until the system converges (which it must, because it is bounded above by the size of the original KB). This model allows us to explore a number of interesting questions, including: (1) How does the growth in the number of facts affect reasoning performance? (2) How might the speed at which different kinds of concepts are learned vary, and what factors does that depend upon? (3) Is learning focused, or are we learning facts about a wide range of predicates and

[4] http://research.cyc.com

concepts? (4) What are the properties of different learning strategies? (5) How does the distribution of facts that can be acquired affect the learning trajectory? This approach provides a general way to explore the evolution of knowledge bases in learning systems. Given the assumptions discussed later, we find that (1) the size of the KB rapidly converges, (2) the growth is limited to a small set of concepts and predicates, spreading to only about 33% of the entire growth possible, (3) different concepts show different rates of growth, with the density of facts being an important determining factor, and (4) Different learning strategies have significant differences in their performance, and the distribution of facts that can be learned also plays an important role. The results imply that as learning systems learn thousands or even hundreds of thousands of new facts from the Web, they will need to be cognizant of the properties of their learning strategies as well as the distribution of facts in the external knowledge source. We conclude that a set of learning methods are probably needed to achieve balanced learning.

- **Growth Patterns of Inference:** In this part, we model the growth of Q/A performance in large knowledge-based learning systems. As discussed above, due to considerable research in Learning by Reading and Machine Reading systems, we have systems which are good at accumulating large bodies of ground facts (although learning general quantified knowledge is currently still beyond the state of the art). But will the new ground facts learnt by them help in improving deductive reasoning? Ideally, new facts should lead to improvements in deductive Q/A coverage, i.e. more questions are answered. Will the rate of performance improvement always be uniform, or will there be "phase changes"? Understanding the dynamics of inference is important to answering these questions, which in turn are important for making self-guiding learning systems. Our approach draws upon ideas from network analysis, where the networks are the AND/OR connection graph of a set of first-order Horn axioms. By analogy to epidemiological models, we explore diffusion of inference in the network, i.e. how does coverage of queries increase as new ground facts are learned. Cascade conditions correspond to when inference becomes easy, i.e. increased coverage. Here we argue that some useful insights about growth patterns of inference can be derived from simple features of search spaces. We focus on three parameters: The first, α, associated with each node, represents the contribution of each node in answering a set of questions. Two other parameters k and β, represent the connectivity of the graph. We study Q/A performance for different values of these parameters, including several sizes of KB contents, to simulate the impact of learning. We found that search spaces with skewed degree distribution lead to better Q/A performance in smaller KBs, whereas in larger KBs more uniform search spaces perform better. In some cases, as α increases, the percolation of inference shows a significant and abrupt change. A degenerate case, in which the effect of ground facts "dies down" and expected improvements in Q/A performance are not observed due to mismatch of expectations and ground facts, is also seen.

- **Modeling the Coordination of Learning Actions in Knowledge-based Systems**: In the last part of this work, we study the importance of coordination of learning actions in knowledge-based systems. Optimal utilization of new facts acquired from the Web (or other sources) could change the face of modern AI systems. However, we must make sure that the benefits of these new facts show up in better Q/A performance. Inundating a KB with irrelevant facts is hardly useful. Therefore, a rational learning system should try to formulate small number of learning goals which would help it to answer more questions. Which learning goals should be selected to maximize Q/A performance? Techniques for selecting a small number of queries are also needed for active learning systems, which interact with a human expert or crowds to augment their knowledge. Since it would be impractical to seek thousands of facts from a single human user, learning systems must limit the scope of their queries. Even with crowdsourcing, selecting queries that avoid gathering irrelevant information is important. Here, we argue that an arbitrary selection of queries would result in large scale unification problems and the effect of new facts would not reach the target queries. We show that the selection of queries for maximizing Q/A performance is similar to a coordination game. We then use reinforcement learning to solve this problem. We model the dynamics of a learning system which sends learning goals to an external knowledge source, and experiments show that this coordination-based approach helps in improving Q/A performance. The results entail that the dependencies of search space induce a small partition (for each query) of the entire domain, which is selected by the reinforcement learning algorithm.

1.2 Organization

Chapter 2 provides the relevant background on our inference engine and Cyc KB.

Chapter 3 describes our axiom extraction method. We discuss our network-based model and study some systemic properties of the KB. Two heuristics are introduced and their efficacy is assessed.

In chapter 4, we discuss how the distribution of facts in KB can be used to prune some axioms and queries. We introduce our static analysis method and show that it helps in improving Q/A performance.

Chapter 5 describes how we can reduce our dependence on axioms. We show how graph-based reasoning can help us in deriving plausible answers. Knowledge patterns written in high-level language help us in limiting our attention to more likely predicate combinations. We then use reinforcement learning to learn the plausibility of knowledge patterns.

In chapter 6, we introduce the inverse ablation model for modeling the evolution of learning systems. The performance of two learning strategies is evaluated.

Chapter 7 describes a model for studying the growth of Q/A performance in large knowledge-based learning systems. We use the inverse ablation model discussed in chapter 6 to model a learning system, and evaluate a diffusion model for spread of inference in deductive search spaces.

In chapter 8, we argue that the process of acquiring facts from an external knowledge source should be seen as a coordination game.

We conclude and discuss future work in chapter 9.

2. Background

While the techniques discussed here are designed to be fully general, we use terminology from Cyc [Matuszek et al 2006] in this paper since that is the major source of the knowledge base contents used in these experiments. We summarize the key terms here.

Cyc represents concepts as *collections*. Each collection is a kind or type of thing whose instances share a certain property, attribute, or feature. For example, `Cat` is the collection of all and only cats. Collections are arranged hierarchically by the `genls` relation. `(genls` *`<sub> <super>`*`)` means that anything that is an instance of <*sub*> is also an instance of <*super*>. For example, `(genls Dog Mammal)` holds. Moreover, `(isa` *`<thing> <collection>`*`)` means that <*thing*> is an instance of collection <*collection*>.

Predicates are also arranged in hierarchies. In Cyc terminology, `(genlPreds` *`<s> <g>`*`)` means that <*g*> is a generalization of <*s*>. For example, `(genlPreds touches near)` means that touching something implies being near to it. The set of `genlPreds` statements, like the `genls` statements, forms a lattice. Because queries involving `genls` and `genlPreds` involve only single antecedents and no new variables, they can be implemented quite efficiently via graph search [Brachman & Levesque 2004], a property we use in this work.

Restrictions on the types of entities that can be used as arguments to predicates are well-known to be useful for reasoning, but as shown below, they are also useful in optimizing sets of axioms. In Cyc terminology, (argIsa <*relation*> <*n*> <*col*>) means that to be semantically well-formed, anything given as the <*n*>th argument to <*relation*> must be an instance of <*col*>. That is, (<*relation*>……<*arg-n*> …) is semantically well-formed only if (isa <*arg-n*> <*col*>) holds. For example, (argIsa mother 1 Animal) holds.

ResearchCyc can be viewed either as incorporating higher-order logic or as a first-order knowledge base with extensive reification. We take the latter perspective here. We use Cyc's predicate type hierarchy extensively. `PredicateType` is a collection of collections and each instance of `PredicateType` is a collection of predicates. The predicates in a given predicate category represented in the KB are typically those sharing some common feature(s) considered significant enough that the collection of all such predicates is useful to reify. Instances of `PredicateType` include `TemporalPartPredicate`, `SpatialPredicate`, `Goals-Attitude-Topic`, `PhysicalPartPredicate` and `PropositionalAttitudeSlot`.

The particular motivation for our work is reasoning within large-scale learning systems like Learning Reader [Forbus *et al* 2007]. Learning Reader was designed to learn by reading simplified texts. The texts were about world history, particularly the Middle East, including its geography, history, and information about current events. Given this initial focus, we developed following *parameterized question templates* [Cohen *et al* 1998] for testing the system's knowledge.

1. Who is <*Person*>?
2. Where did <*Event*> occur?
3. Where might <*Person*> be?
4. What are the goals of <*Person*>?
5. What are the consequences of <*Event*>?
6. When did <*Event*> occur?
7. Who is involved in <*Event*>?
8. Who is acquainted with (or knows) <*Person*>?
9. Why did <*Event*> occur?
10. Where is <*SpatialThing*>?

In each template, the parameter (e.g., <*Person*>) indicates the kind of thing for which the question makes sense. Each template expands into a disjunction of formal queries. For example, question 3 uses queries involving the predicates `hasBeenIn`, `citizens`, and `objectFoundInLocation`. (We included these predicates in the experiments described below as one set of queries.). Learning Reader used a subset of the contents of ResearchCyc as of 2007, consisting of 1.2 million axioms. It was able to learn by reading, as measured by improvements in its scores on quizzes consisting instances of these questions. In addition to using deduction for question-answering, it also used deduction in *rumination*, an off-line process where the system mulled over what it had read by posing its own questions and trying to answer them for itself.

For convenience in reading, all predicates in Cyc begin with lower case letters, while collections and constants begin with a capital letter. Horn clauses are indicated by

(<== $C\ A_1\ A_2\ \ldots\ A_n$)

where A_1, A_2, …,A_n,are the antecedents and C is the consequent. For example, one of the queries for question template 8 where <*Person*> was given as `BillClinton` would be `(acquaintedWith BillClinton ?x)`. Backchaining on this query leads to an answer in which ?x is bound to `HillaryClinton`, using these axioms:

```
(<== (acquaintedWith ?x ?y) (spouse ?x ?y))
(<== (spouse ?x ?y) (wife ?x ?y)).
```

Our approach to tractable inference is to restrict backchaining to small sets of axioms, automatically constructed from the general clauses of the knowledge base, that are optimized for particular classes of queries. We call these axioms sets *chainers*. Chainers correspond to a single partition in the sense of [Amir *et al* 2005]. Axioms in chainers are restricted to definite Horn clauses, and are not range restricted. Using Horn clauses sacrifices completeness for efficiency. In most applications, it is better to fail quickly and try another approach than to pursue a query for hours or days and then fail. The Cycorp HALO system, for example, would have dramatically improved its score by adding a 30 second time-out for queries, since no query taking longer than that was ever successful.

In the next chapter, we have used DAMP [Martin & Reisbeck 1986] for parsing some text. In DMAP, the parsing algorithm is a process of lexically-guided memory search in which predictive patterns of words and concepts guide a general memory search process to recognize relevant memory structures [Martin & Reisbeck 1986].

3. Automatic Extraction of Efficient Axiom Sets for Commonsense Reasoning in Large Knowledge Bases

3. 1 Introduction and Motivation

Deductive reasoning is an important component of many AI tasks. Efficient first-order reasoning systems can be built today only for small-to-medium sized knowledge bases and by careful hand-tuning of inference mechanisms and representations. There are two reasons to seek more general solutions. First, hand-tuning and careful crafting do not scale as the size of the knowledge base grows. For example, queries that fail in large knowledge bases frequently take hours to fail.[5] There is still no evidence that general-purpose first-order reasoning in such knowledge bases can regularly be performed in order of a few minutes per query in today's computing environments. The second problem is that knowledge bases are no longer being built entirely by hand. Advances in machine learning and knowledge capture provide the opportunity to automatically construct large knowledge bases [Etzioni et al 2005, Forbus et al 2007]. But to exploit this opportunity, we must be able to reason with them effectively.

It is well-known that knowledge representation choices play a crucial role in determining the hardness of problems. Work in SAT problem solving is moving towards understanding the structure of problems and using it for the design of better heuristics [Kautz & Selman 2007, Walsh 1999]. First-order knowledge bases, though structured, are highly complex networks of concepts. Predicates are connected through different types of relationships, defining intricate networks. Understanding KB structure is fundamental to many important knowledge representation and reasoning problems. These include evaluating inference engines and assessing the effectiveness of heuristics and algorithms.

Here we exploit properties of the structure of a large knowledge base to improve inference. This chapter is organized as follows:

[5] cf. www.projecthalo.com/content/docs/

- We describe the *ExtractAxioms* algorithm, which automatically constructs an efficient set of Horn clauses from a first-order knowledge base consisting of general clauses. It takes as input a set of target queries, information about the current distribution of ground facts in the KB, and information about the kinds of facts that can be provided in the future (for example, via machine learning or knowledge capture or learning by reading).
- We propose that for understanding reasoning performance, the search space represented by the axioms in the knowledge bases should be seen as a network where the nodes are predicates mentioned in axioms, and the edges connect the predicates mentioned in the antecedents to the predicate of the consequent. Given such a representation, we observe that in such a graph, most nodes have very few children, whereas some nodes have many children.
- We improve the performance of *ExtractAxioms* algorithm by using these topological properties of predicate connectivity. We use the distribution of known facts and what might be added to identify difficult and less useful regions of the KB, and use that information to prune inefficient axioms, which improves performance.
- We propose that the maximum out-degree of nodes in the search graph is a key parameter for controlling the navigability of search spaces. Networks which have high degree nodes, called *hubs*, are inherently unsuitable for focused search. We study the family of search spaces, from disconnected to scale-free, by varying this parameter and show that it helps in identifying efficient sets of axioms.
- We demonstrate that these techniques are useful by showing how they improve performance in queries from multiple sources, including queries from an existing learning by reading system and examples from the Thousands of Problems for Theorem Provers (TPTP) corpus.

3.2 Motivating Tasks and Approach

As [Forbus et al 2007] describes, the Learning Reader prototype has been shown to learn from reading simplified English texts. Two sets of axioms extracted via these techniques were essential to the system's performance. A small chainer was extracted for rapid reasoning during question answering. A larger chainer was extracted for use during rumination, i.e., the offline process by which the system poses itself questions and tries to answer them, as a way of improving its understanding of what it had read. An automatic analysis of the phrasal patterns used by the DMAP parser provided information about the types of predicates that might be obtained by reading, making it useful to estimate the utility of axioms not just on current KB contents, but also on what can be provided via reading.

For practical AI systems, reasoning directly with quantified knowledge (i.e., at least first order) is essential. Brute-force techniques like pre-computing all ground facts (propositionalization) are infeasible for two reasons. First, it leads to combinatorial explosions and hence does not scale for large knowledge bases. For example, consider just one axiom from the KB:

```
(implies (and (doneBy ?X ?Z) (eventOccursAt ?X ?Y))
   (holdsIn ?X (objectFoundInLocation ?Z ?Y)))
```

There are more than 14,000 ground assertions which match the pattern `(eventOccursAt ?X ?Y)`, 5333 ground assertions that match the pattern `(doneBy ?X ?Z)`, leading to 1056 consequences. And this is only one axiom. Second, the propositionalization of axioms involving logical terms can lead to infinite sets of statements. Third, propositionalization assumes that the set of facts is static, i.e. that a set of facts is identified once, in advance, and never changes. This does not match the needs of many tasks, including systems that deal with changing environments and that learn.

Of course, the problem of determining whether a set of first-order Horn clauses entails an atom is undecidable. As usual, finding reasonable completeness/efficiency tradeoffs is a prime concern, especially as the size of the knowledge base grows. In the next section, we show how to construct chainers that provide efficiency with little loss of completeness.

3.3 Extracting Efficient Sets of Axioms

Here we describe our ExtractAxioms algorithm for extracting axioms from the KB (see Figure 3.1). We have observed that ground facts are not uniformly distributed across predicates. Consequently, this method is based on the idea that while searching, we should focus on regions of the search space that (i) are rich in ground facts or (ii) involve facts that can be produced by external systems (e.g., machine learning, learning by reading, or knowledge capture systems). In this algorithm, we represent the predicates used in statements that can be produced by external systems by the set *LearnablePredicates.* If a predicate P belongs to the set *LearnablePredicates* then ExtractAxioms would include axioms with P in the antecedent even if it is not currently very frequent in the KB. We assume that the set of

LearnablePredicates can be constructed by examining the structure of the system that is producing facts. (If the external system is producing new predicates, then by assumption they cannot be in the knowledge base already, and hence are irrelevant for this algorithm.) However, it is unlikely that the distribution of statements using particular predicates produced by external systems will be known in advance, and consequently, we include all such predicates in *LearnablePredicates.*

We focus on a single query predicate without loss of generality, since selecting axioms for a set of queries can be done by taking the union of axioms generated for each predicate. The essence of the algorithm is a backward sweep from the query predicate through Horn clauses extracted from general KB axioms involving that predicate. The extraction process constructs Horn axioms from general first-order axioms, assesses the efficacy of all Horn clauses and outputs those Horn axioms which would be most useful in answering questions.

It is well-known that depth cutoffs are very important for adjusting completeness versus efficiency in reasoning. Therefore, we include a depth cutoff as one of the parameters for our algorithm. In the ExtractAxioms algorithm, *KnownFacts*(p) represents the number of ground facts about the predicate *p*. We define *InferredFacts*(p) as the sum of ground facts of all nodes below *p* in the `genlPreds` lattice. InferredFacts takes into account the answers that can be produced by genlPreds inference, which is always performed because, as noted in Section 2, it can be done efficiently. Formally it is $\sum_X$ *KnownFacts*(x), where X is the set of predicates reachable from p via `genlPreds` links pointed downward i.e., to more specific predicates. We define *AllFacts(p)* as *KnownFacts(p)* + *InferredFacts(p)*, i.e., the total number of times a predicate *p* could be proved via ground facts and `genlPreds` inference. For example, *AllFacts*(`performedBy`)= 392, which means that we can get 392 answers for the query `(performedBy ?x ?y)` if we restrict the reasoning to ground fact retrieval and `genlPreds` inference. For example, the predicates `senderOfInfo`, `driverActor`, `failureForAgents`, and `seller` contribute 194, 2, 7 and 2 facts respectively.

In step 2 of Figure 3.1, we create a simplified backward cycle-free search graph from Horn clauses that can conclude the predicate *pred*. The graph is simplified because we are interested in identifying the predicates which could be the bottlenecks. For a rule s(?x, ?y) ← p(?x, ?y) ^ q(?x, ?y), we create three nodes labeled *p*, *q* and *s*. Directed links exist from *s* to *p* and from *s* to *q*. (Note that rules like s(?x, a)← p(?x, b) ^ q(?x, c) and s(d, ?y) ← p(e, ?y) ^ q(f, ?y) will lead to the same graph.) For each node *p*,

```
Algorithm ExtractAxioms:
Input:
    • pred: A predicate for which axioms are to be extracted
    • depth: depth cutoff, typically 5
    • LearnablePredicates: A set of predicates which can be generated by an external
      system, such as a learning system.

Output: A set of axioms, SelectedRules, for proving the predicate pred.
    1. SelectedNodes ← Ø, SelectedRules ← Ø
    2. Make a backward search graph, T, until depth = depth, by converting axioms
       mentioning pred into Horn clauses, and recursing on their antecedents.
    3. Convert T to a queue, by performing a topological sort to order the nodes of the
       directed graph.
    4. Threshold ← 0.001* $\sum_X$ KnownFacts(x) where X is the set of all predicates in
       the graph.
    5. Repeat step 6 until T is empty.
    6. Pop an element y, from the ordered list T and include it in SelectedNodes if
          a. y ε LearnablePredicates or,
          b. y is an OR node and Children(y) ∩ SelectedNodes ≠Ø or,
          c. y is an AND node and Children(y) is a subset of SelectedNodes or,
          d. AllFacts(y) > Threshold
    7. SelectedRules ← { r | r is a Horn clause in T and all the predicates in its
       antecedents ε SelectedNodes.}
    8. Return SelectedRules
```

Figure 3.1: ExtractAxioms Algorithm used to extract axioms from KB.

Children(p) represents the children of *p* in the search graph. We would like to include all paths from the frequently provable nodes to the root. To achieve this, we perform a topological sort of the nodes and begin from the leaves. A predicate is chosen if it is in *LearnablePredicates* or if the number of ground statements using it is higher than a given threshold[6]. Since this is a backward search graph, we include the parents of selected nodes (steps 6(b) and 6(c)). In step 6(d) we prefer those regions of KB which are rich in ground facts. The set *SelectedNodes* represents the nodes which can be frequently proved. In step 7, a Horn clause is included in *SelectedRules* if all predicates in its antecedents are in *SelectedNodes.*

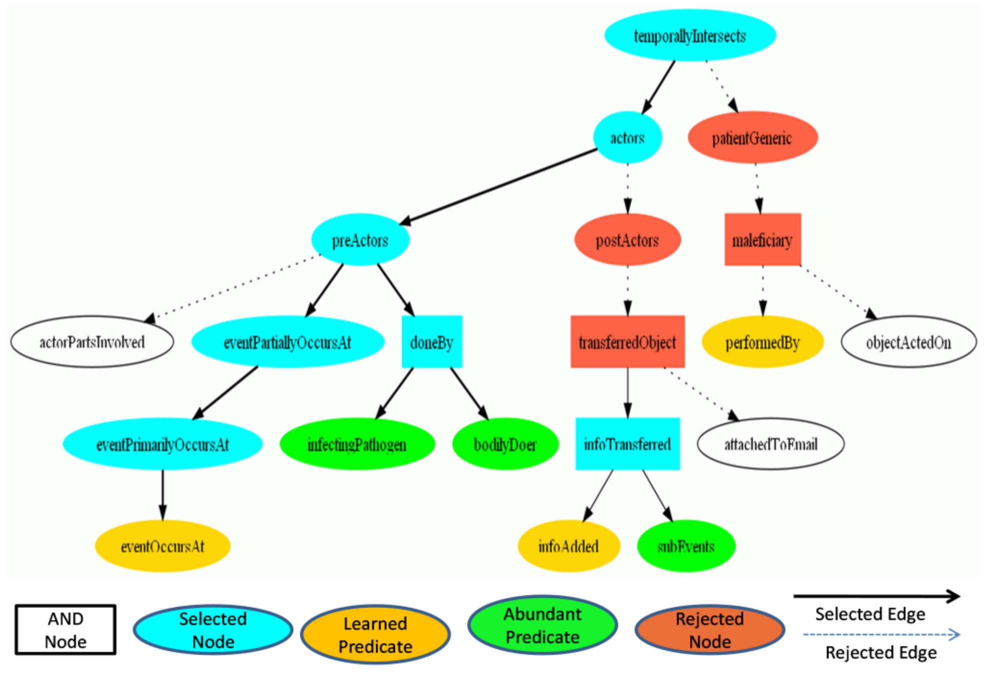

Figure 3.2: A Simplified illustration of ExtractAxioms algorithm.

In Figure 3.2, we show a simplified example of how the algorithm works for the predicate `temporallyIntersects`. Square nodes represent AND nodes in the search space[7]. Uncolored nodes

[6] The threshold is set to 0.1% of the sum of number of ground facts of all predicates in the search graph. When this threshold is high, condition 6(d) is not easily satisfied and reasoning with *LearnablePredicates* is preferred by the algorithm. This threshold is sufficient for answering queries within 90 seconds. It is clear that when the threshold is 0, all predicates with at least one ground fact are selected. Very few axioms would be selected if the threshold is set to 1% of the sum of number of ground facts of all predicates in the search graph.
[7] The search space is an AND/OR query graph, where unification is needed at AND nodes.

like `objectActedOn` (see Figure 3.2) do not have many ground facts associated with them in the KB and they cannot be produced by the reading system either. Therefore, the whole search path from `objectActedOn` to `patient-Generic` is rejected. The nodes, which satisfy the condition in step 6(d) of the algorithm, are shown in green. The nodes shown in yellow and green are selected as well (see step 6(a) and 6(d) in Figure 3.1).

Let us consider another concrete example. Axiom 3.1, shown below, would not be chosen by this method because there are not many statements or axioms which could help in proving the antecedent `(hasEmotionAboutType ?ARGS-1 ?ARGS-2)`. On the other hand, axiom 3.2 would be selected because there are over 14,000 statements which match the pattern `(eventOccursAt ?x ?y)`.

```
(←(dislikesType ?ARGS-1 ?ARGS-2)
  (dislikes-Generic ?ARGS-1 ?ARGS-2)
  (hasEmotionAboutType ?ARGS-1 ?ARGS-2))
```
(Axiom 3.1)

```
(← (actors ?x ?y)
    (eventOccursAt ?x ?y))
```
(Axiom 3.2)

Could this method be fooled by adding thousands of irrelevant facts (e.g., incorporating a phone book)? This is not possible, because the relevance of facts is determined by the axioms that mention them. For example, any system that contains a substantial amount of natural language knowledge will have predicates like nameString, which describe how some conceptual entity might be described linguistically. Generally a predicate like nameString would be involved in reasoning about names, and would not appear in a search graph for causal queries. Continuing this example, 97.5% of axioms in our KB involving nameString have a string indexing predicate in their consequent, e.g.

```
(←(trademarkStringOf ?AGENT ?PRODTYP ?STRINGX)
   (makesProductType ?AGENT ?PRODTYP)
   (nameString ?AGENT ?STRINGX))
```
(Axiom 3.3)

The complexity of ExtractAxioms is quite reasonable. Let K and N be the set of axioms and predicates respectively. Moreover, let E be the set of edges of the graph. Then, the complexity of computing

InferredFacts(p) is $O(|N|^2)$. Step 2 requires $O(|N|.|K|)$ time. Topological sort requires $O(|N|+|E|)$ which is $O(|N|^2)$. Step 5 and 6 are $O(|N|^2)$ too. Therefore, the complexity of pre-processing step is $O(|N|^2)+O(|N|.|K|)$. Since the axiom extraction process occurs off-line, and infrequently compared to the number of times that the axioms are used, this is a very reasonable cost. Next, we use the structure of the KB to further improve the performance of axioms. Then in Section 4, we study the efficiency of reasoning using the sets of axioms produced by these techniques.

3.4 Knowledge Bases as Networks

Many queries can take long time to fail. Consequently, we want to use heuristics to identify and remove the bottlenecks. We describe some topology-based and distribution-based heuristics that can provide significant improvement in time/performance tradeoffs.

As noted above, it is useful to think about large knowledge bases as networks, where the nodes are predicates and links exist from the predicate of the consequent of an axiom to the predicates in the antecedent. For concreteness, this analysis uses the ResearchCyc KB, but we suspect that similar properties will hold for any large, general-purpose KB. We focus on the predicates involved in queries for the questions discussed in Chapter 2. As mentioned previously, we built a simplified backward search graph until depth 5 for the predicates involved in question templates shown in Chapter 2. This graph had 4,864 nodes. Given our notion of connectivity, the cumulative degree distribution of nodes is shown in Figure 3.3. We observe that most of the nodes have very few links, whereas a few nodes have very high degree. This distribution is highly right skewed and resembles a power law distribution[8]. For example, 2,300 predicates have no children i.e. their out-degrees are zero, while 12 predicates have out-degrees greater than 100. One predicate (`isa`) had 854 neighbors, the highest in our experiments. Similarly, other predicates like `temporallyCoexists` and `genls` had 358 and 149 neighbors respectively. Numerous studies have shown that such networks are ubiquitous in natural phenomenon. It has been found that many networks, including the world-wide web, a cell's metabolic systems, and Hollywood actors, are dominated by a small number of nodes that are connected to many others [Mitchell 2006]. This distribution suggests that the knowledge in ResearchCyc uses a small set of predicates heavily. Our ability to infer useful facts hinges on them. If these predicates are known, inference is surprisingly easy.

[8] In its most general form, a power law distribution has the form $p(x) \alpha x^{-a}$, where $2<a<3$ although there are occasional exceptions. In such heavy tailed distributions, the low frequency population tails off asymptotically.

On the other hand, inferring them can be difficult. Such non-uniform distributions are amenable to targeted intervention or perturbation, which we can exploit to improve inference.

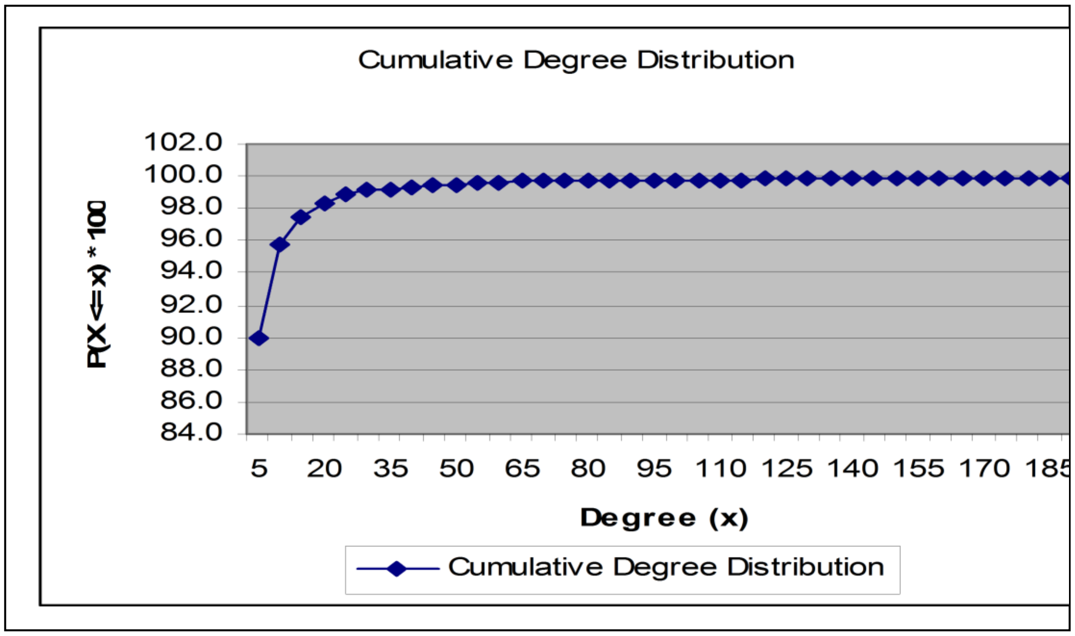

Figure 3.3: Degree Distribution

Consider the connectivity of the network as we remove nodes. Clearly, the more nodes we remove, the more likely it would be to fragment the network. Random networks collapse after a critical number of nodes have been removed. However, since the connectivity of scale-free networks depends on a small number of hubs, random failures cannot disconnect them. This extreme robustness is accompanied by fragility to attacks: the systematic removal of a few hubs would disintegrate these networks, breaking them into non-communicating islands. [Jeong et al 2001]

These properties have significant implications for making reasoning more efficient. Our heuristics are based on the intuition that nodes with high degree are queried for repeatedly and can be proved in many ways. This ensures that these predicates need significant time to fail[9]. One option is to remove these nodes i.e. stop reasoning about them. However, removing high-degree nodes disconnects the network and the number of questions answered drops significantly. At this stage, very little chaining takes place and questions are answered only by ground-fact retrieval. Therefore, we need to keep a minimum number of

[9] We note that this problem cannot be solved by using breadth-first search. If the answer lies at depth 5, then a high degree node at depth 2 would remain a bottleneck for both depth-first and breadth-first strategies.

such nodes to ensure that we answer a reasonable number of questions. To do this, we use the distribution of known and inferred facts. If a predicate is frequently known, then the hardness of search space below it is less relevant because it is less likely that we will need to derive them via reasoning. Moreover genlPreds inference is typically easier than reasoning with normal axioms[10]. Therefore, the function *AllFacts(p)*, defined above, provides a good measure for identifying predicates which could be easily and frequently proved. We use it to exclude those high-degree predicates which do not have a high value of *AllFacts(p)*. This heuristic is referred to as the *Hub Removal Heuristic* (See examples in Figure 3.4 and 3.5). Our results in Section 3.4 show that this heuristic can help us to get an efficient set of axioms.

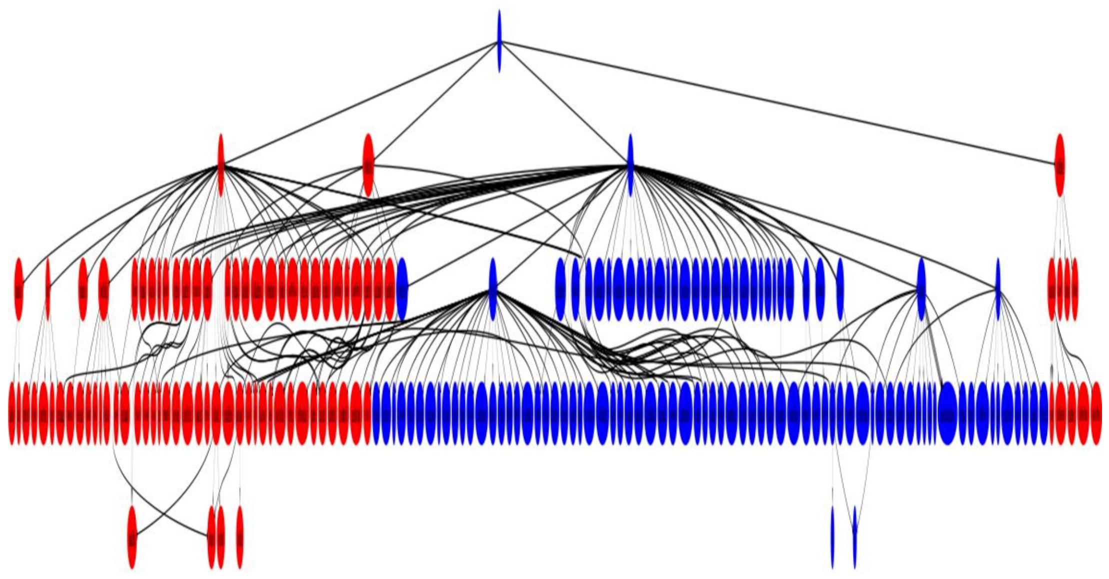

Figure 3. 4: Simplified illustration of *Hub Removal Heuristic*. High degree nodes in ground fact rich regions are accepted (The accepted sub-tree has been colored blue). Other high-degree nodes and the sub-tree below them are removed (colored red). A concrete example is shown in Figure 3.5.

[10] The efficacy of an axiom decreases with the number of antecedents. Axioms for `genlPreds` reasoning have just one antecedent.

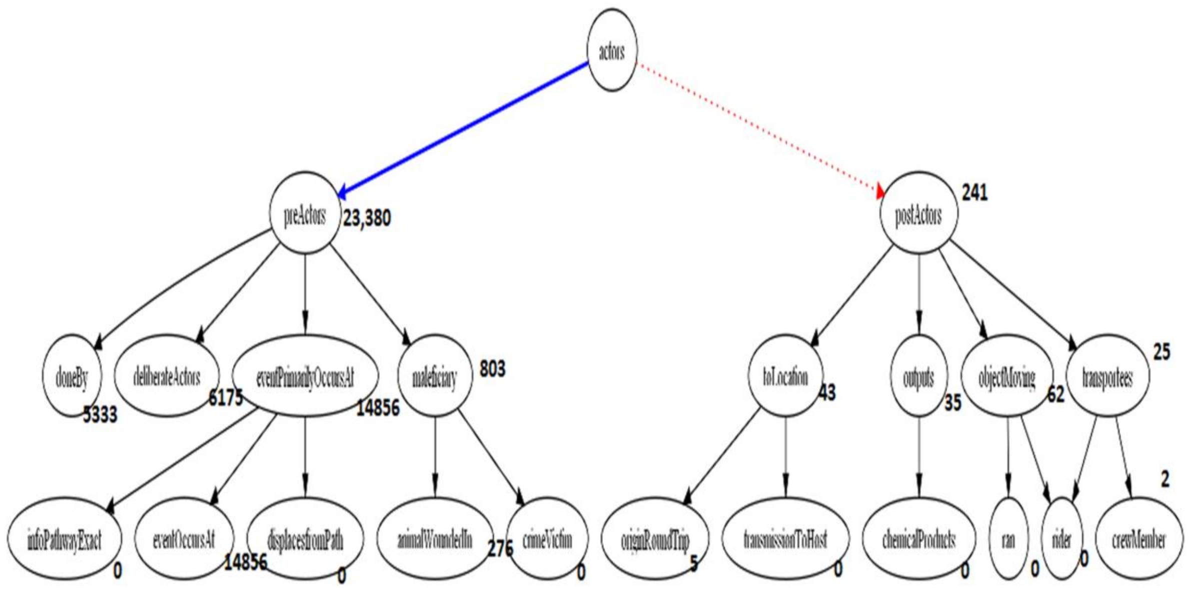

Figure 3.5: A simplified example of *Hub Removal Heuristic*. The numbers show the value of AllFacts(p) for the predicates. Let us assume that preActors and postActors have significantly higher out-degree than others. Here preActors has more ground facts in the sub-tree below it than postActors. Therefore, the link between actors and preActors would be preferred over the link between actors and postActors.

However, we can take another approach for solving the same problem. Networks which have hubs have a diameter which is bounded by a polynomial in log(N), where N is the number of nodes. In other words, there is always a very short path between any two nodes [Kleinberg 2000]. Once the reasoning process reaches one of these hubs, most of the remaining network is easily reachable. For example, in our experiments, we have observed that most of the predicates are accessible during search through predicates like `isa`, `genls` and `holdsIn`. This makes the search intractable and queries are timed out. Therefore, one possible solution is to prevent the network from having hubs[11]. We can do this by limiting the maximum out degree of each node in the search space. Let m be the maximum allowed out-degree of each node. If V is the set of vertices in the resulting graph, then by definition $m = \max_{v} \deg^{-}(v)$. In other words, we do not allow any node to have an out-degree greater than m. When $m = 0$, the graph is a set of disconnected nodes. On the other hand, when $m = \infty$, the out-degree is not limited and we have the original graph. Between these two extremes, we can study the ease of navigability of a family of search spaces. When m is low, short paths between nodes do not exist. As we increase m, the connectivity of graph improves. After a particular threshold, there are too many potential paths in the graph and relevant

[11] In this analysis, we consider a hub to be a node which has more than 50 children. Less than 1% of the nodes satisfy this condition.

paths are difficult to find. This threshold determines the optimal complexity of the search space. In the next section, we use *AllFacts(p)* to restrict *m*, and to determine its optimal value. This is similar to Kleinberg's notion that efficient navigability is a fundamental property of only some networks and after a critical threshold individuals (or algorithms) cannot easily find paths in social networks [Kleinberg 2000]. We call this heuristic the *Uniform Graph Heuristic*. In Figure 3.6 and 3.7, we show how the parameter *m* changes the nature of search space. Figure 3.6 shows a part of the graphical version of the output of *ExtractAxioms* algorithm (Here the out-degree of nodes is not restricted, therefore m is ∞.). For simplicity, the node labels and direction of edges have not been shown. In Figure 3.7, we show the optimized version of the same region of search space when *m* is set to 3.

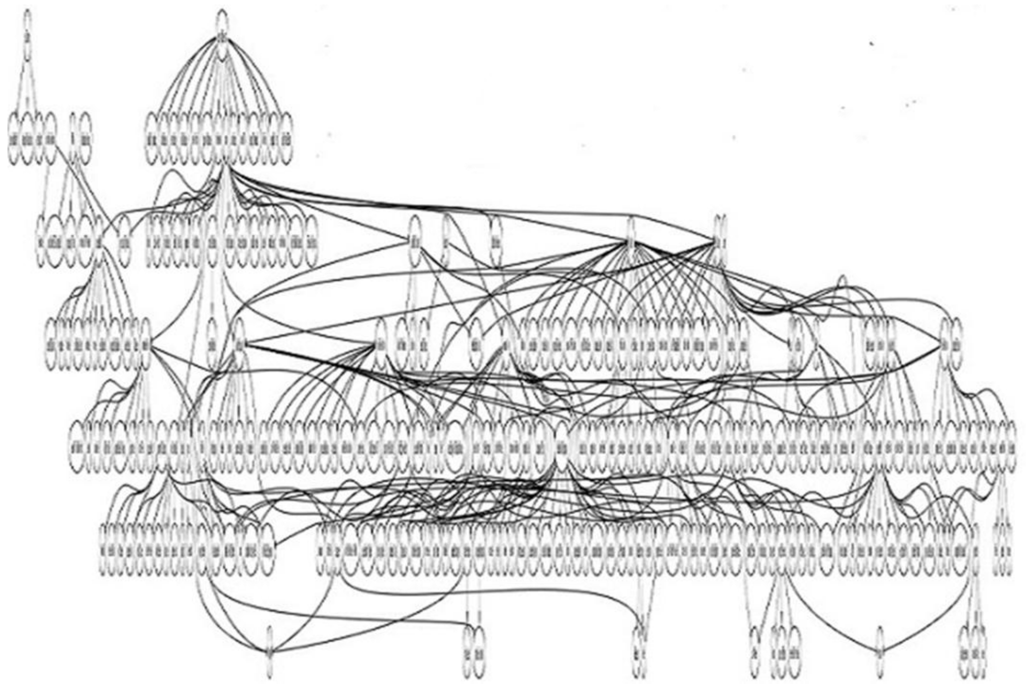

Figure 3.6: Simplified Graphical view of a part of the search space represented by the output of ExtractAxioms algorithm. Here *m* is ∞, which means that there is no limit to the number of children a node can have. See optimized version in Figure 3.7.

In the next section, we use *AllFacts(p)* to restrict *m*, and to determine its optimal value. This is similar to Kleinberg's notion that efficient navigability is a fundamental property of only some networks and after a critical threshold individuals (or algorithms) cannot easily find paths in social networks [Kleinberg 2000]. We call this heuristic the *Uniform Graph Heuristic*. In Figure 3.6 and 3.7, we show how the parameter *m*

changes the nature of search space. Figure 3.6 shows a part of the graphical version of the output of *ExtractAxioms* algorithm (Here the out-degree of nodes is not restricted, therefore m is ∞.). For simplicity, the node labels and direction of edges have not been shown. In Figure 3.7, we show the optimized version of the same region of search space when *m* is set to 3.

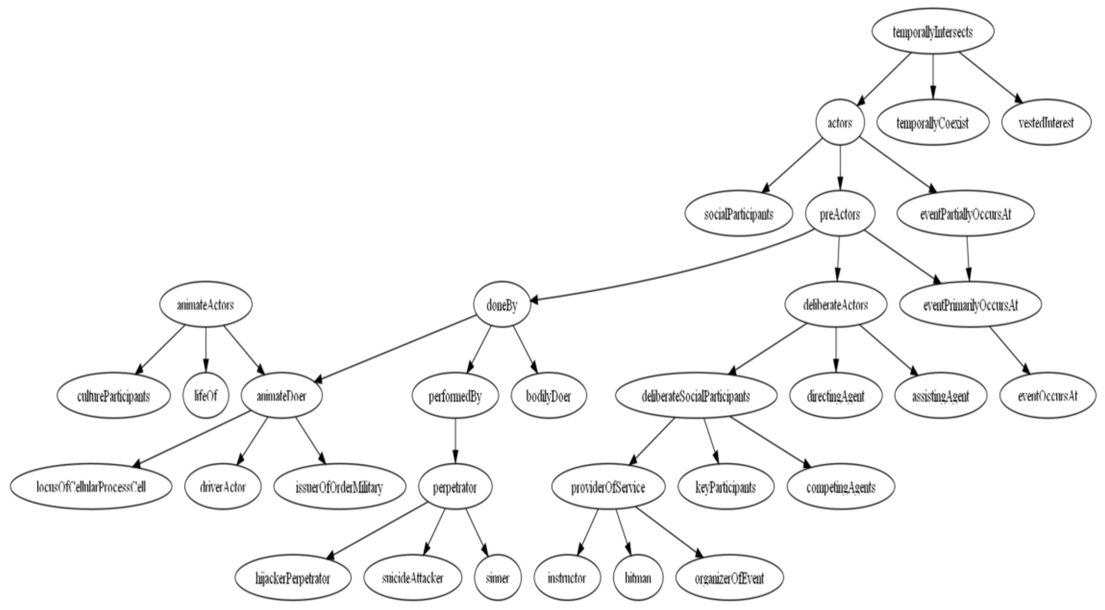

Figure 3.7: After pruning with the *Uniform Graph Heuristic*. This is the optimized version of search space shown in Figure 3.6. Here *m* is 3, which means that each node can have at most 3 children. Regions of the knowledge base which are rich in ground facts are preferred.

3.5. Experiments

To illustrate the utility of these ideas, we describe a series of experiments using three sets of questions (Table 3.1). The first corpus of questions (**Q1** henceforth) was generated by randomly selecting entities from the KB satisfying the question templates, creating 100 questions of 10 parameterized question-types, for 1,000 questions total. The second corpus of 970 questions (**Q2** henceforth) was constructed by automatically generating all legal instantiations of the parameterized questions for the entities constructed by the overall system's reading in the experiment described in [Forbus et al 2007]. We also wanted to check that our methods worked for other predicates. To do this we sorted all predicates by *AllFacts(p)*. We then selected 61 top predicates and replaced their first argument by 100 randomly sampled entities satisfying their argument constraints[12]. This led to a set of 6,100 queries. This set of questions is referred to as **Q3** in the following section. The set *LearnedPredicates* for question set **Q2** was initialized by the set of predicates in DMAP patterns. It was initialized to the empty set for other question sets. Each query was timed out after ninety seconds. Allocation of more time for queries does not help due to the size of the search space. If the inference engine cannot find the solution in first few seconds, then the size of irrelevant search space ensures that no solution can be found. We used a simple backchainer [Russell & Norvig 2003] working with a logic-based truth-maintenance system [Forbus & de Kleer 1993]. We sought one instantiation of the answer and searched until depth 3. All experiments were done on a 3.2 GHz Pentium Xeon processor with 3GB of RAM. Here we evaluate how the topology of the network

Name of Query Set	**Source of Question Templates**	**Source of Entities for instantiating the Question Templates**	**Predicates in *LearnedPredicates***	**Number of Queries**
Q1	Learning Reader [Forbus et al 2007] Question Templates	Random selection from the KB	Ø	1000
Q2	Learning Reader Question Templates	Entities mentioned in the interpretations of Learning Reader Corpus	Predicates from the phrasal patterns of DMAP [Martin & Reisbeck 1986]	970
Q3	Top 61 predicates with highest number of ground facts in the genlPreds lattice below them.	Random selection from the KB	Ø	6100

Table 3.1: Question Sets

affects the time taken to answer questions. (In all graphs we study the percentage of questions answered or time taken to answer them.) For the query sets **Q1** and **Q2**, we begin with all predicates in the question

[12] The predicates which have high value of *AllFacts(p)* are very general predicates and have many specializations. Since we answer questions by backchaining, their specializations are automatically considered for selection during the axiom extraction process.

templates discussed in Chapter 2. For **Q3**, we begin with the set of 61 predicates discussed earlier. We make a search tree for these predicates and use the algorithm ExtractAxioms to get the axiom set EXA. The set EXA for **Q1, Q2** and **Q3** had 6,350, 6,417 and 7,894 axioms respectively. For a baseline comparison, we use the set of all Horn clauses for all predicates in the search tree.

Notation	**Description**
Baseline	All Horn axioms for all predicates until depth 5 in the search tree.
EXA	Output of algorithm ExtractAxioms
HighDegreeNodes$_x$	Sort the predicates on the basis of their out degree and keep x% with the highest out degree
MostKnownPredicates$_x$	Sort the predicates on the basis of AllFacts(x) function and keep x% with the highest value.
MostKnownNeighbors(v, x)	Begin with EXA. Make the backward search tree, sort the children of *v* via AllFacts(p), keeping the top *x* children.
Consequent(i)	Predicate in the consequent of an axiom *i*
Antecedents(i)	Set of Predicates in the antecedent of an axiom, *i*
ACD$_x$	{ i ε EXA | Antecedents(i) \ MostKnownNeighbors (Consequent(i), x) = Ø}
AHD$_i$, i>0	EXA\ All Rules which mention predicates in HighDegreeNodes$_i$, in their antecedents.
AMK$_i$, i >0	EXA\All Rules which mention predicates in (HighDegreeNodes$_i$ \ MostKnownPredicates $_i$) in their antecedents.

Table 3.2: Notation

First, we would like to understand the impact of high degree nodes. We study this relation by removing all axioms which mention high degree predicates in their antecedents. In Table 3.2, HighDegreeNodes$_x$ represents top x% of high degree nodes. The set, AHD$_i$ (**A**xioms without **H**igh **D**egree predicates), is obtained from EXA by removing all axioms which mention predicates in HighDegreeNodes$_i$ in their antecedents.

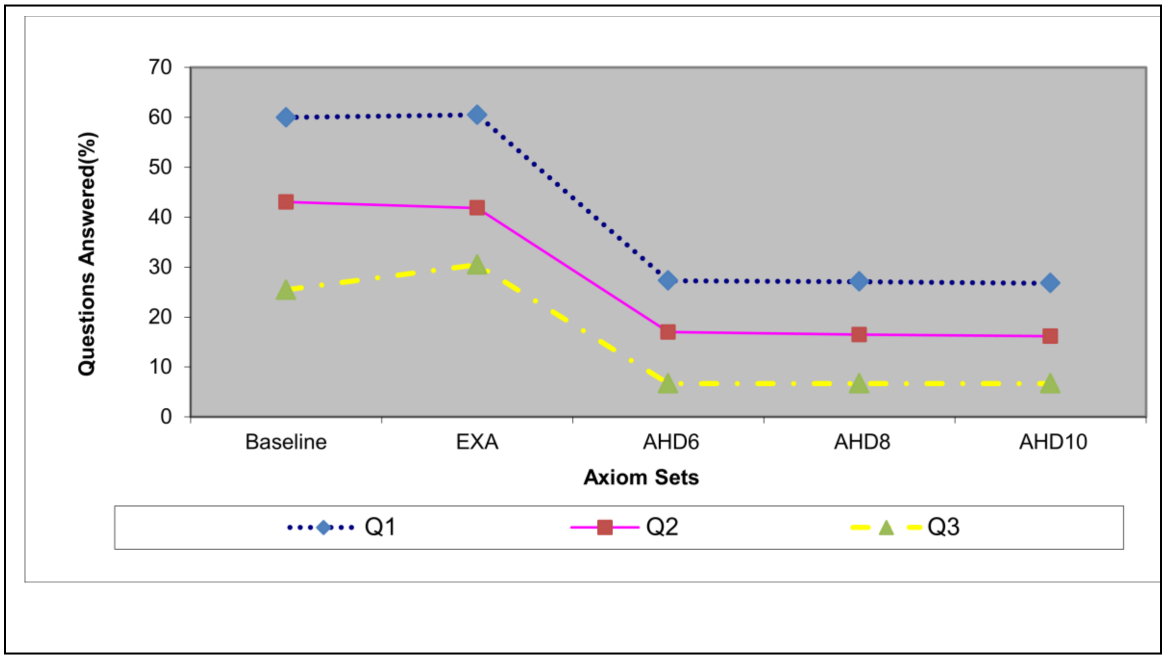

Figure 3.8: Effect of removal of high degree nodes on completeness

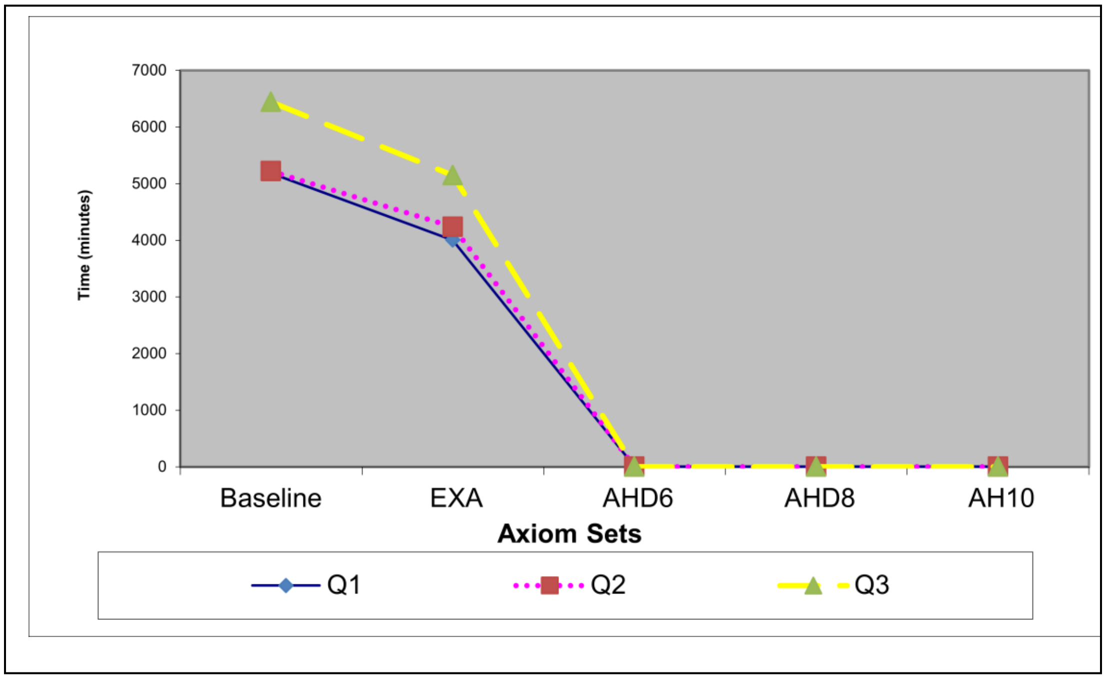

Figure 3.9: Effect of removal on high degree nodes on time required.

For example, to get HighDegreeNodes_2, we sort the predicates on the basis of their out degree and keep the top 2%. Then AHD_2 is obtained by removing all axioms from EXA which mention predicates in HighDegreeNodes_2. We see in Figure 3.8 that as we remove high degree predicates, the network falls apart[13]. (Different sets of axioms are plotted on the x-axis.) Virtually no inference is possible, and coverage is significantly affected. All questions answered at this stage are by retrieval and minimal chaining takes place. Moreover, the number of questions answered remains same for AHD_6, AHD_8 and AHD_{10} which suggests that the search space had been fragmented to a set of non-communicating islands for AHD_6 and removing more predicates did not cause any change. Figure 3.9 shows that time required for AHD_6, AHD_8 and AHD_{10} is very close to zero which provides additional evidence that the search space is disconnected.

This evidence suggests that removing a small number of high degree nodes significantly decreases time requirements at the cost of recall. However, we need to include some high degree nodes to improve question-answering performance. Next we use the distribution of ground facts for this purpose. The set $\text{MostKnownPredicates}_x$ represents x% of predicates which have the highest number of ground facts in the specialization hierarchy below them. In our model, this set represents predicates which should not be excluded even if they have higher degree than others. The set $\text{HighDegreeNodes}_x \setminus \text{MostKnownPredicates}_x$ represents the predicates which are in ground fact poor regions of the KB and have many children in the search space. In Table 3.2, we remove those axioms from EXA which have these predicates in their antecedents to get the set AMK_x (**A**xioms with **M**ost **K**nown Predicates).

[13] EXA is the output of algorithm ExtractAxiom algorithm (Fig 3.1). There is significant drop in the Q/A performance when we remove top 6% nodes with the highest out-degree (compare the performance of EXA and AHD_6).

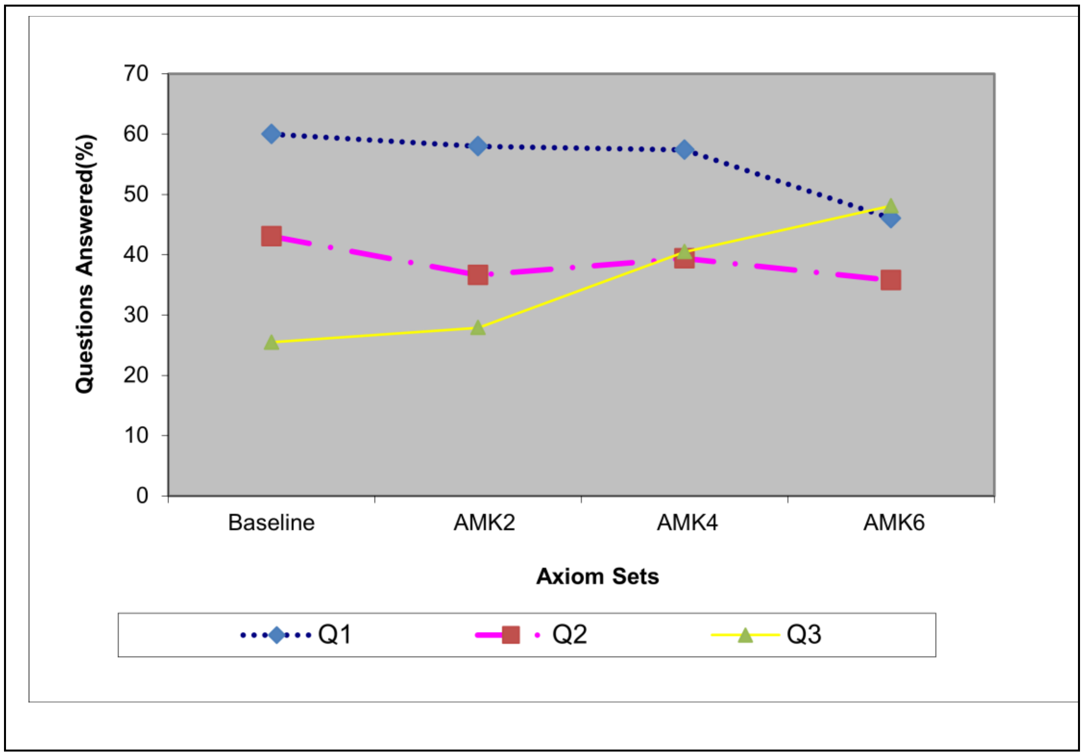

Figure 3.10: Effect of inclusion of Most Known Predicates on completeness

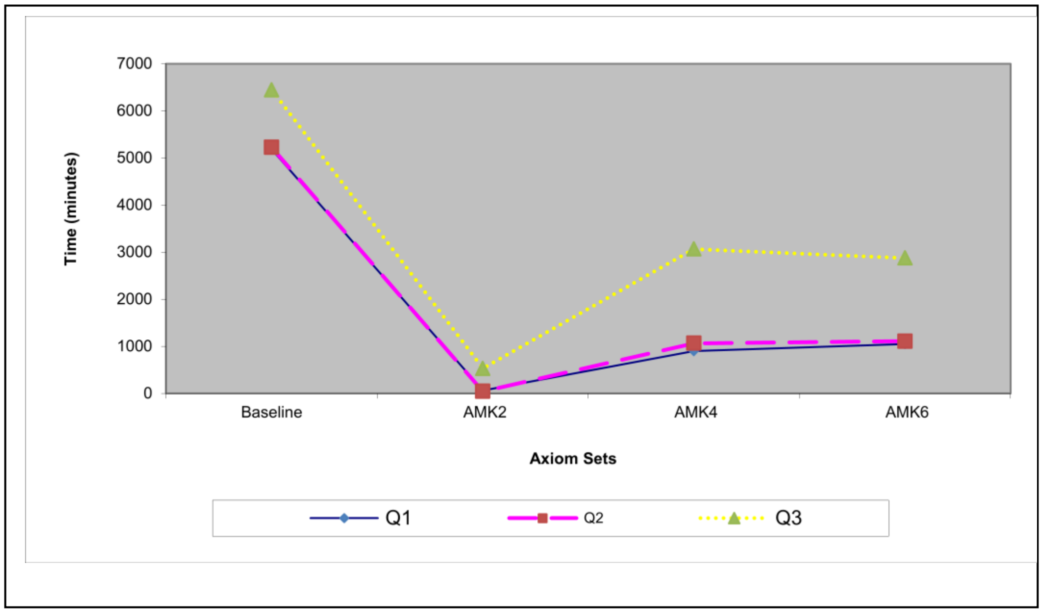

Figure 3.11: Effect of inclusion of Most Known Predicates on time requirements.

We study the performance of AMK_x in Figure 3.10 and 3.11. Figure 3.10 shows that, by including the predicates in $MostKnownPredicates_x$, we have recovered most of the losses shown in Figure 3.8. Figure 3.11 shows that there is significant improvement in time requirements. This heuristic leads to a factor of 80, 4.90 and 2.24 improvement for the Q1, Q2 and Q3 query-sets respectively. The maximum loss in completeness is 8.5%. This method is referred to as **'Hub Removal Heuristic'** in Table 3.3.

Next we show that maximum out degree is a reasonable parameter for modeling efficient navigability of the network. In Section 3.3, we defined *m* as the maximum out-degree of nodes in the graph. When $m=\infty$, we have the original rule set or EXA. To remove bottlenecks, we would like the network to be more uniform. The set MostKnownNeighbors(v, x) is obtained by sorting the children of the predicate v by *AllFacts(p)* and selecting the top *x* from them. The set ACD_x (**A**xioms with **C**onstrained Maximum **D**egree of nodes) is obtained by removing all axioms whose antecedents are not in the set MostKnownNeighbors(v, x) where v is the predicate in the consequent of the axiom. In the search space represented by ACD_x, none of the nodes have more than x children[14].

We study the performance of this heuristic in Figure 3.12 and 3.13. In Figure 3.12, we observe that the optimal set generated from this heuristic performs better than the baseline and EXA in all cases (See Table 3.3 and Figure 3.8 for baseline numbers). Moreover, we get significant improvements in time when we impose the maximum-degree condition (Figure 3.13), i.e., use the Uniform Graph Heuristic. We also observe that the question answering performance worsens significantly in at least one case (see performance for the query set Q3 in Figure 3.12) when the maximum degree is increased beyond 18. This provides evidence that too many edges hinder quick navigation towards the target.

The final results are shown in Table 3.3. We note that ExtractAxioms' output, EXA, significantly improves the performance compared to the baseline. It improves the performance by 22%, 18% and 20% for Q1, Q2 and Q3 sets respectively. The maximum loss of completeness is 2.7%. Both heuristics further improve the performance of this set. The set of axioms which led to best performance are shown in parentheses. We see that in most cases we have been able to improve the performance significantly. The Hub Removal Heuristic might lead to some incompleteness but it leads to significant time savings. The Uniform Graph Heuristic always leads to improved performance without any loss of completeness.

[14] These heuristics reduce the number of axioms significantly. For example, ACD_3 and AMK_4 reduce the number by 67% and 50% respectively for the question set Q3.

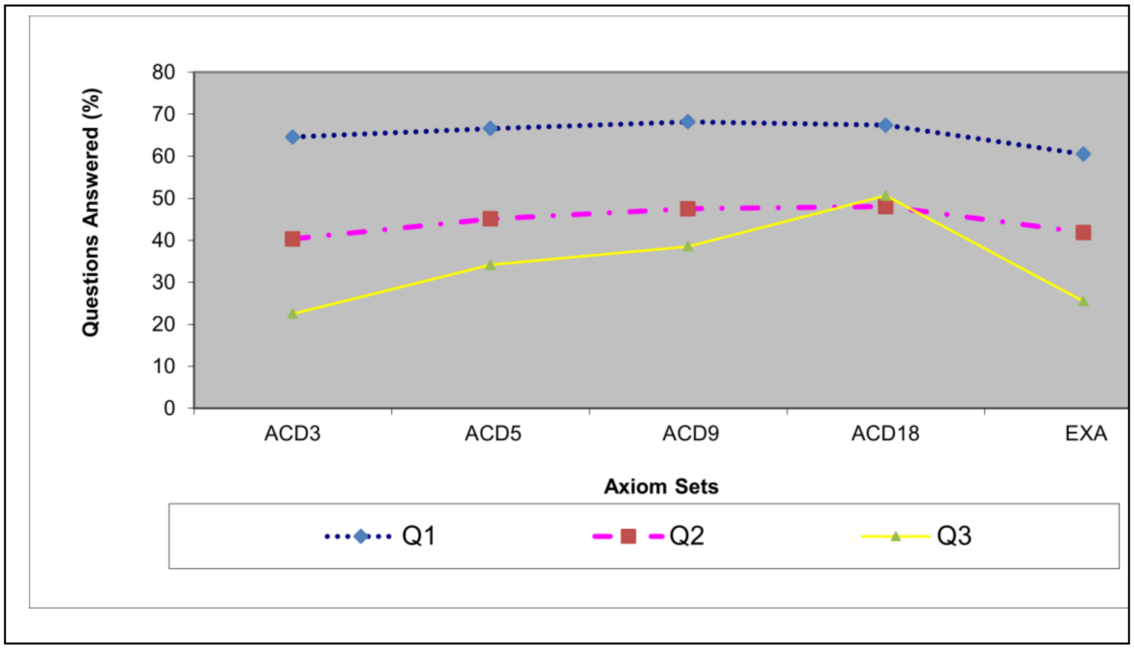

Figure 3.12: Effect of imposing maximum degree condition on completeness

It might seem counterintuitive that we can answer more questions faster by removing rules. Returns diminish because as we increase the number of rules, many queries get lost in less useful regions of search space and are timed out. We verified this hypothesis by increasing the timeout for the EXA set for the Q3 set of questions. The results are shown in Table 3.4. We see that the improvement is not encouraging. In fact, our heuristics outperform it even though those queries are allowed less than 120 seconds. In other words, allowing more time simply increases failure time and does not help in answering queries. We show the scaling of ExtractAxioms with depth in Figure 3.14 and 3.15. We conclude that performance tapers off after a threshold and providing more resources is not useful.

Query Sets	**Rule sets**	**% Answered**	**+/- %**	**Time (min)**	**Speedup**
Q1	Baseline	60.00	0%	5183.70	1
	EXA	60.50	0.83%	4005.39	1.3
	Hub Removal Heuristic (AMK_2)	58.00	-3.33%	64.00	81
	Uniform Graph Heuristic (ACD_3)	64.60	7.67%	7.78	666
Q2	Baseline	43.05	0%	5222.96	1
	EXA	41.86	-2.76%	4240.55	1.2
	Hub Removal Heuristic (AMK_4)	39.37	-8.55%	1064.76	4.9
	Uniform Graph Heuristic (ACD_9)	47.50	10.34%	273.54	19.1
Q3	Baseline	25.49	0%	6444.21	1
	EXA	30.50	19.65%	5150.39	1.2
	Hub Removal Heuristic (AMK_6)	48.16	88.94%	2874.02	2.2
	Uniform Graph Heuristic (ACD_{18})	50.63	98.63%	1568.48	4.1

Table 3.3: Summary comparison of performance

Rule Set EXA Timeout	Q3(%)	Q3(minutes)
30 sec.	23.98	2076.65
60 sec.	27.91	3769.80
90 sec.	30.50	6444.21
120 sec	32.77	6496.51

Table 3.4: Effect of increasing timeouts on performance

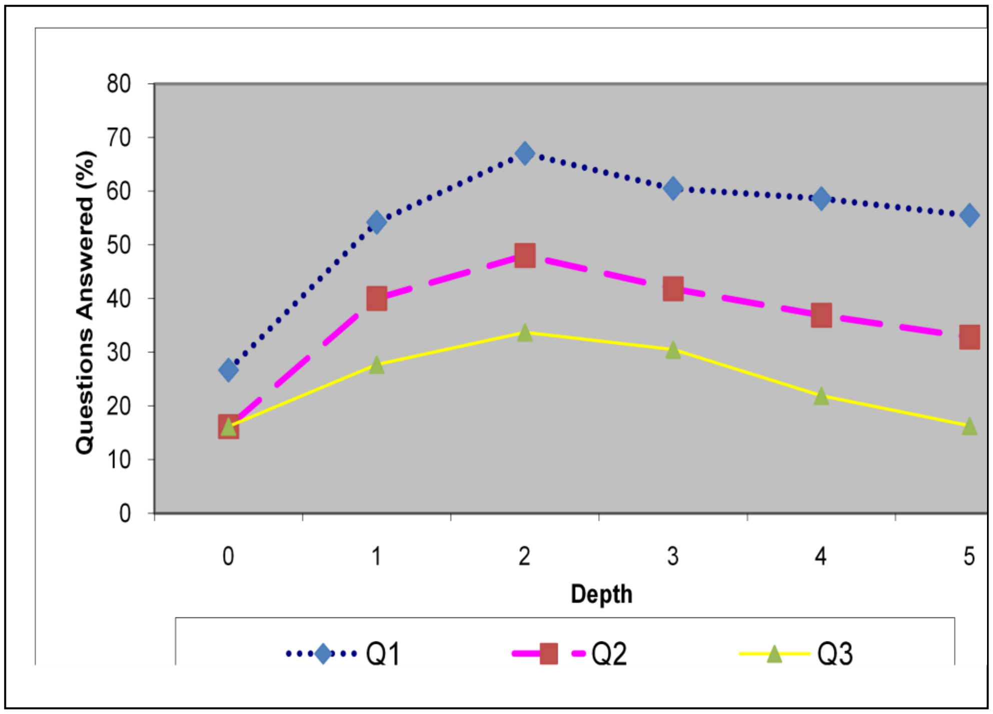

Figure 3.14: Effect of *depth* on performance

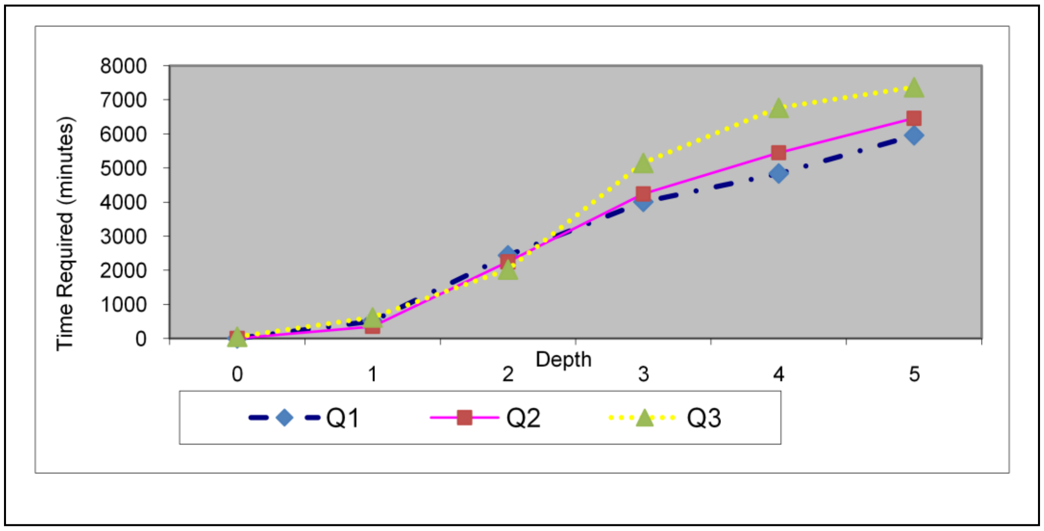

Figure 3.15: Effect of *depth* on time required to answer questions.

3.5.1 TPTP Experiments

We also evaluated our techniques on several sets of problems from the Thousands of Problems for Theorem Provers (TPTP) data set [Sutcliffe 2009]. These experiments were divided in two categories. Type 1 problems were from TPTP Cyc scaling problems set (CSR026+1 to CSR074+6). These problems are among the largest problems available in the TPTP problem set, so performance on them is of interest. However, they still use the Cyc representation. In order to see how our methods perform on non-Cyc representations, we constructed a second set of problems (the Type 2 problems), consisting of 50 satisfiable problems from the following domains:

- The Agents Domain (AGT)
- Natural Language Processing Domain (NLP)
- The Management Domain (MGT)

The 50 problems were chosen by eliminating small problems and weeding out close isomorphs.

We chose these specific domains because they included large problems and also included irrelevant axioms. Many TPTP domains are designed to stress particular features of theorem provers, and hence virtually all of the given axioms are relevant[15]. The Type 1 and Type 2 problems contain irrelevant axioms, thus challenging systems to separate the wheat from the chaff.

The baseline was obtained by solving the original problem using the full set of Horn clauses extractable from the given clauses. Our experiments have shown that the performance of Uniform Graph Heuristic is better than the Hub Removal heuristic. Therefore, we used the former (ACD_5) for solving these problems and measured the speedup. Each query was timed out after 20 minutes. We sought one instantiation of the answer and searched until depth 5. All experiments were done on a 3.2 GHz Pentium Xeon processor with 3GB of RAM.

The results are shown in Table 3.5. The performance for Type 1 problems is better because the work presented here is best suited for optimizing larger problems.

	No. of problems attempted	% solved successfully	Speedup
Type 1	294	81%	11.2
Type 2	50	88%	4.0

Table 3.5: Experimental Results for problems from TPTP problem set

The TPTP problems solved here are different from our other experiments in three ways: (a) The TPTP problem set is best suited for checking the strengths of contradiction checking procedures. State of the art theorem provers have made great progress in improving the proof by contradiction techniques. However, commonsense reasoning in large KBs typically involves proof by construction, which requires very different optimization approach [26]. The proof by contradiction approach works efficiently for small problems. They have, however, faced different problems while solving larger problems (e.g., not being able to even load a Cyc-sized knowledge base, due to memory issues [Ramachandran et al 2005]). For

[15] For example, ANA001-1, English: "A continuous function f in a closed real interval [a,b] attains its minimum (or maximum) in this interval."

instance, recent experiments have shown that when there were 534,435 formulas in the problem set, two theorem provers solved 12% and 0% problems respectively [Morbini & Schubert 2009]. We note that the KB used for non-TPTP experiments discussed in this section had 1.2 million facts. (b) As mentioned above, efficient reasoning in many problems involves choosing the correct axiom at a given stage of solution construction. In such problems (e.g., problems in Group theory), all axioms are considered relevant. Therefore, *a priori* assessment of efficiency of axioms is not possible. (c) Finally, problems in TPTP consist of a set of facts followed by a single query or conjecture. Therefore, the theorem prover is not expected to be cognizant of the tradeoffs involved in answering a set of queries from a given KB. In contrast, our approach supports large scale learning systems like Learning Reader [Forbus et al 2007]. In such systems, the Q/A module is expected to answer multiple classes of questions (e.g., `(eventOccursAt <Event> ?x)`) over a changing knowledge base. Statistically, the proofs of such queries are more likely to use the ground fact rich regions of the KB. Therefore it is reasonable to try to identify ground fact rich and relevant sections of KB, and choose axioms which access these parts of the KB. However, the TPTP problem set is more of a set of regression tests than a general knowledge base intended for multiple uses. In other words, since only one proof path is relevant for each problem, it can be "hidden" in an obscure part of the KB. This does not trouble us because in large scale learning experiments, such queries have empirically been rare.

3.6 Applicability to other Query Sets

How can we use these ideas in other systems? As mentioned above, the performance of Uniform Graph Heuristic is better than the Hub Removal heuristic. (A closer look at Table 3 shows that Uniform Graph heuristic always outperforms the Hub Removal heuristic.) Next, we need to find the best value for Uniform Graph heuristic's parameter (i.e., the maximum allowed degree of nodes). Figure 12, 13 and Table 3 provide us some hints about the relation between the maximum allowed degree of node and the improvement in performance. It is clear that as we increase the maximum allowed degree of nodes, the optimized set of axioms will tend towards the unoptimized set. It follows that we will get comparatively less improvement in time requirements. Therefore, our initial strategy should be to keep the value of this parameter low. Selecting a small number of children for each node without sacrificing Q/A performance is only possible if the ground facts needed for answering the target queries are concentrated in a small region of the search space. In such a case, choosing a small number of neighbors for each node would cover all/most useful regions. An example of this "easy reasoning regime" is the query set Q_1 discussed above. What happens if the search queries are disparate? In such a case, the ground facts needed for answering a significant number of queries would lie in many different corners of the KB and no small "set cover" would be found. To ensure good Q/A performance, we will need to choose a large number of neighbors for each node. It follows that there would be small improvement in time requirements. An example of this "hard reasoning regime" is the query set Q_3 in our paper. Note that the queries in Q_3 are not similar to each other.

Therefore, given a new KB and new query types, we will need to find if the queries are similar and the knowledge of KB is concentrated in small region. In such a case, we should use ACD(x), where x is low (i.e., $x < 9$). On the other hand, if the queries are dissimilar and the facts are distributed uniformly across the KB, then we should use ACD(x), where x is high (i.e., $x > 9$).

3.7 Related Work

Complexity issues have driven many reasoning researchers to restrict themselves to propositional theories. For example, propositional Horn clause theories have been used to approximate more complex propositional theories, to take advantage of the linear-time complexity of propositional Horn reasoning [Selman & Kautz 1996]. Another example is the SAT reasoning community [Kautz & Selman 2007], which has extensively investigated the problem of optimizing propositional reasoning. Other researchers have developed techniques for converting first-order problems into propositional reasoning problems. For example, [Ramachandran and Amir 2005] present an algorithm for converting reasoning problems in monadic first order logic into equivalent problems in propositional logic. Similarly, answer set programs are frequently compiled to propositional logic in order to use SAT solvers [Baral 2003]. Such techniques can be effective on small sets of axioms, but lead to combinatorial explosions that make them impractical for large knowledge bases. Our results suggest that automatic construction and optimization of first-order Horn axioms is quite practical and already operates at scale on large knowledge bases. [Lang *et al* 2003] discuss syntactic and semantic forms of independence for a propositional KB. Such results are not relevant here for at least three reasons. Firstly, we are using first-order logic and propositionalization is not feasible. Secondly, given the nature of axioms, it is very unlikely that the types of independence relations discussed there would be useful for our domain. Finally, most (in)dependence relations discussed by them have a high complexity.

Despite our focus on first-order theories, there are commonalities in our approach with some of the optimization ideas that the SAT and CSP communities have been developing. In SAT solving, the fixed clause models of hardness, where the ratio of clauses to variables is considered to determine the hardness of the problem, has received considerable attention [Mitchell et al 1992]. Recently, heavy tails in randomized search costs have been acknowledged as a serious problem. Non-uniform distribution of search-costs points towards the fact that not all variables are equally important. In fact, such behavior has frequently been explained in terms of *backdoors* and *backbones*. [Gomes et al 2004, Kilby et al 2005] The idea is that different groups of variables in a problem encoding often play quite distinct roles. For example, a plan search technique that branches purely on the independent variables can obtain substantial speedups over search methods that do not exploit variable dependencies [Kilby et al 2005]. Our analyses identify similar structure for logical first-order KBs. Evidence from an analysis of the performance of state of the art theorem provers [Morbini & Schubert 2009, Ramachandran et al 2005] indicates that commonsense reasoning in large knowledge bases involves tackling quite different issues than these systems have been optimized for. Our work is complementary to work on variable ordering strategies, removal of redundant constraints and identifying backtrack free graphs in CSP, connection-graph based techniques [Kowalski 1975, Stickel 1982] and theorem-proving literature [Manthney & Bry 1988] because we propose heuristics for simplifying the problem structure and quickly identifying where answers could be. Any inference engine should be able to benefit from our work. Our work is closer to [Walsh 1999], who showed that graphs of real world problems are not uniform but have a 'small-world' structure. The idea of branching factor has been used in parsing literature [Charniak 1986]. To the best of our knowledge, there has not been any work in the AI community which has studied the correlation between network structure and time/performance tradeoffs in deductive reasoning.

3.8 Conclusions

As knowledge bases grow, especially via machine learning, learning by reading, and knowledge capture, better ways to automatically use such knowledge efficiently must be found. This chapter describes two techniques for this important problem. The first is to automatically extract subsets of the KB targeted at particular tasks, exploiting knowledge of what kinds of statements are available already and what might be supplied via other systems, such as learning systems. The second uses an analysis of the connectivity of knowledge bases to automatically identify nodes to prune which, while losing a small amount of completeness, can yield over an order of magnitude performance improvement. The average speedup in our large KB experiments is a factor of 129. The worst case is a factor of 4 improvement in time with only 8.5% loss in completeness. In many cases, we have been able to improve completeness over an exhaustive baseline. The results on 344 problems from TPTP problem set are also encouraging.

As noted in Section 3.2, these techniques have already been used in Learning Reader, a system that learns by reading simplified English texts. This provides evidence for their utility in systems that integrate reasoning and learning. We believe that they are applicable to a wide range of such systems. For example, information extraction systems use templates from which possible future fact types can be derived.

These results suggest three lines of future work. First, carrying out similar experiments on other large KBs would be informative. Unfortunately, we believe that at present they do not exist. While there are a reasonable number of medium to large scale ontologies in existence, the only knowledge about the terms in them tends to be structural knowledge and ground facts, not general axioms relating the terms in ways that support deep inference[16]. Given the current interest in learning by reading and reading the web, we expect this to change. We hope that the existence of more efficient reasoning techniques like these will help motivate the construction of more large-scale KBs. Second, we think coupling a network-based analysis like ours with other semantic analysis of axioms [Walsh 2003] could yield a more complete theoretical picture as to what makes inference hard. Finally, the ability to identify what makes inference hard potentially provides us with the information as to what might make inference easier – in other words, analyze the structure of knowledge bases to ascertain what kinds of knowledge could be added to improve inference, and thus help generate and prioritize learning goals for systems that accumulate knowledge.

[16] In Learning Reader, we found that at least 5 to 6 axioms (including `genlPreds` statements) are needed for answering interesting queries.

4. Improving Commonsense Reasoning in Large Knowledge Bases by Semantic Static Analysis

4.1 Introduction

Recall that by knowledge base we mean a collection of formally represented knowledge, including logically quantified first-order expressions as well as ground facts. This is in contrast with, for example, Wikipedia, OpenMind [Singh *et al* 2002], or other collections of text which are often informally called knowledge bases. We also mean more than simply ontologies, such as SUMO, which only provide structural relationships between concepts, not general-purpose axioms. Nor do we include lexical resources, which provide information about words (e.g., WordNet, VerbNet, FrameNet) but whose intended semantics is expressed informally in natural language, rather than via axioms that constrain their meaning. While text collections, ontologies, and lexical resources have their uses, none of them directly provide the axioms needed for deductive reasoning.

Today there is only one family of large-scale first-order knowledge bases, the Cyc systems. We believe that there are two reasons for this. The first is that building large-scale knowledge bases by hand requires monumental resources. As noted above, the rise of new means of automatically (or semi-automatically) capturing knowledge is changing this factor. The second reason is that existing deductive reasoning techniques do not scale well, so accumulating more knowledge degrades performance. Existing approaches to scaling rely on some means of selecting subsets of the knowledge base to use for particular tasks. For example, axioms in Cyc are organized into *microtheories*, contexts which have inheritance relationships specified between them, thus defining a logical environment for reasoning that is a subset of the entire knowledge base (e.g. Lenat & Guha, 1989). While experiments have been done to automatically place new facts into microtheories [Taylor et al 2007], the overall organization of the microtheory structure is still done by hand, rather than automatically. Another approach is to compute partitions of the KB dynamically, based on connectivity of axioms (e.g., Amir & McIlraith 2005). While we believe both of these techniques are important, we have found them insufficient for scaling deductive inference.

This chapter describes how reasoning engines can improve their performance by exploiting statistical properties of knowledge bases. These properties are calculated via an off-line static analysis, and provide estimates of the likelihood of success for particular axioms and search paths. We present three approaches. (1) We describe a method for analyzing axioms and ground facts to approximate the *parameterized deductive closure*, i.e. an estimate of kind of things which could be proved when inference is restricted to depth *d*. This database provides a model of system's knowledge that is used to prune queries which are not likely to be useful, thereby improving performance. (The pruned queries are also potentially useful in determining knowledge goals for learning.) (2) we show that, frequently, the antecedents of axioms jointly impose very strong constraints on the types of entities that can satisfy them, which can be used to select whether or not an axiom will be useful in a specific query. Although the axioms we use are quantified, first-order knowledge, we are inspired by constraint processing methods in the propositional CSP literature [Mackworth 1977, Dechter 2003, Mezard *et al* 2002]. We use similar ideas to develop the notion of the *global constraint space*, which is much smaller than constraints induced by predicates. Our experiments show that reasoning in this space leads to an order of magnitude improvement in time and space with no loss in completeness. (3) Finally, we show that proofs generated over a training set of queries can be used to estimate the likelihood of an axiom contributing to a successful proof. These likelihoods are used to prune less useful rules. Experiments show that such an approach leads to significant savings in time and memory.

4.2 Motivation

Commonsense reasoning is challenging because it involves reasoning about vast amounts of knowledge, flexibly, while at the same time being quite rapid. For AI systems that perform commonsense reasoning, reasoning directly with quantified knowledge (i.e., at least first order) is essential. Brute-force techniques like pre-computing all ground facts (propositionalization) are infeasible because of the size of KB. For example, consider the following axiom from the ResearchCyc KB:

```
(implies (and (doneBy ?X ?Z) (eventOccursAt ?X ?Y))
  (holdsIn ?X (objectFoundInLocation ?Z ?Y)))
```

There are more than 14,000 ground assertions which match the pattern `(eventOccursAt ?X ?Y)`, 5,333 ground assertions that match the pattern (doneBy ?X ?Z), leading to 1,056 consequences. With 10^6 axioms, this is clearly impractical, and even more so when one considers the number of facts that can be harvested via information extraction from the world-wide Web.

4.3 Estimating Deductive Closure

The difficulty of reasoning depends on the structure and contents of the knowledge base. Since the information in KB is not uniformly distributed across all predicates and concepts, some queries are more likely to succeed than others. This section shows that the distribution of information in the KB can be used to predict the outcomes of queries, which in turn can lead to significant performance improvements. The fundamental insight is that axioms impose strong constraints on what types of entities can be successfully bound to their variables. It is well-known that type constraints associated with predicates can often be used to quickly rule out impossible queries [Frisch 1987, Lenat & Guha 1989]. However, we have observed that the actual distribution of entities is significantly smaller than the distribution of entities that satisfies the argument constraints. By using knowledge of distribution of the ground facts in the KB, we can pre-compile an estimator that detects queries which have virtually no chance of being successful. Pruning such queries can lead to an order of magnitude in time savings.

Consider the query `(maleficiary AAAI-Conference-2000 ?x)`. Since the argument constraint for the first argument position is `Event`,this query cannot be predicted to fail based on type information alone. However it is extremely unlikely that this query would succeed: Conferences tend to not be held to harm someone or something. A KB may or may not contain the appropriate general axioms to allow such an inference to be made. However, analyzing the distribution of ground facts could reveal that instances of Conference never have such information known about them, and hence any such query can be marked as false.

Which factors are important for estimating the probability that a query will be proven, given the distribution of facts and available resources? Depth of inference is a commonly used measure, since it is implementation-independent but extremely relevant, since search time typically grows exponentially with depth. Therefore we estimate the likelihood of proving a query when inference is restricted to depth *d*.

As argued above, another important constraint on inference is the types of the instances involved. Different categories of arguments often involve different axioms. For example, the axioms used for proving `(eventOccursAt G0023 ?x)` will be quite different if `G0023` is a `TerroristAttack` versus a `PhotosynthesisEvent`. Without loss of generality, in what follows we restrict ourselves to queries of the form

```
(<predicate> <instance of collection C> ?x).
```

Let us introduce some definitions that will be useful in estimation:

Definition 4.1: Let **P, C** and **I** be the set of all predicates, collections (i.e., concepts) and entities (i.e., instances) respectively. For example, `performedBy` ε **P,** `TemporalThing` ε **C** and `BillClinton` ε **I**.

Definition 4.2: Let *arity*: **P** → **N** ∪ **{**n-ary**}** be the function that maps predicates to the number of arguments they expect, where **N** is the set of natural numbers. For example, *arity*(`doneBy`) = 2.

Definition 4.3: Let ***GroundFacts****:* **P** × **C** × **N** → **W** be the function that maps a predicate, a collection and an argument position to the number of ground facts involving the predicate in which the entity in the argument position is an instance of the collection. Here **W** is the set of whole numbers. For example, *GroundFacts*(`performedBy, Country, 2`) = 100 represents that there are 100 answers for the query below if we restrict the reasoning to ground fact retrieval.

```
(and (performedBy ?x ?y)(isa ?y Country))
```

Definition 4.4: Let ***SuccessEstimate:*** **P** × **C** × **W** × **W** → **R** be the function that maps a predicate *p*, a collection *c*, an argument position *n*, and a depth *d*, to a likelihood that, given the number of facts involving *p* in the KB, a query involving *p* with an instance of *c* in argument position *n* is answered when the depth of inference is limited to *d.* For example, *SuccessEstimate* (`objectFoundInLocation, Tiger, 1, 3`) = 0.3 represents our belief that 30% queries of the type `(objectFoundInLocation Tiger-111 ?x)` can be answered when depth of inference is limited to 3 and where `Tiger-111` is a randomly chosen instance of the collection `Tiger`.

Definition 4.5: Let **Domain: P × N → $\mathbf{2^I}$** be the function that maps a predicate and an argument position to all instances that satisfy the argument constraint. For example, **Domain**(`objectFoundInLocation,1`) is the set of all instances of `SpatialThing`.

It is also useful to abstract the information available in the KB and assess the information in terms of collections. For example, we can view the domain of predicates in terms of collections.

Definition 4.6: Let **$\mathbf{Domain^c}$**: P × N → 2^C be the function that maps a predicate and an argument position to all collections that are the specializations of the argument constraint. For example, `(FemaleFn Cat),(FemaleFn Dog)` ε `Domain`^{c} `(mother, 2).`

Definition 4.7: Let **Size: C → W** be the function that maps a collection to the number of instances in it. For example, Size(`GeographicalRegion`) = 11594.

Given these definitions, we can calculate SuccessEstimate via the following equations:

$$\boldsymbol{SucessEstimate(p,c,pos,d)} \leftarrow \frac{\sum_{col \in C} \boldsymbol{GroundFacts(p,col,pos)}}{|\boldsymbol{Domain(p,pos)}|}\ ,\ \ when\ d = 0\ and\ c\ is\ unspecified$$

(Equation 1a)

$$\boldsymbol{SuccessEstimate(p,c,pos,d)} \leftarrow \frac{\boldsymbol{GroundFacts(p,c,pos)}}{\boldsymbol{Size(c)}}, when\ d = 0\ and\ c\ is\ specified$$

(Equation 1b)

When the estimate given by the above equations is zero, then pruning those queries is similar to enforcing node consistency in constraint satisfaction literature [Mackworth 1977]. In our context, the node consistency procedure can be written as shown in Figure 4.1.

```
Procedure **Node Consistency** ($P$, $i$)
Input: A Predicate $P$
    Argument Position $i$
    $Domain^c(P,i) \leftarrow Domain^c(P,i) \cap \{c|SuccessEstimate(p,c,i,0) > 0\}$
```

Figure 4.1: Node consistency subroutine

The subroutine shown in Figure 4.1 has to be executed for all argument positions of all predicates. Initially the domain contains all specializations of the argument constraint. For example, consider the second argument position of `mother`.

Domain^c(`mother, 2) =`

`{FemaleHuman, (FemaleFn Cat), (FemaleFn Goat), (FemaleFn Mosquito), …}`

After the execution of subroutine shown above, this domain reduces to {`FemaleHuman`} because the KB does not contain information about the mother-child relationships of non-humans. Therefore, if we only rely on ground facts for answering questions, then queries involving non-humans can be pruned.

Let us consider the following query in which inference (i.e., backchaining) is needed to answer questions. We are expected to estimate the likelihood that the query would be answered from the set of axioms shown below when the depth of inference is limited to 1.

Query: `(eventPartiallyOccursAt Bombing-100 ?x)`

```
Axioms: {A1: (← (eventPartiallyOccursAt ?ATTACK ?LOCATION)
                 (isa ?ATTACK AttackOnObject)
                 (objectAttacked ?ATTACK ?OBJECT)
                 (objectFoundInLocation ?OBJECT ?LOCATION))


      A2: (←  (eventPartiallyOccursAt ?EVENT ?PLACE)
               (subEvents ?EVENT ?SUB-EVENT)
               (eventOccursAt ?SUB-EVENT ?PLACE)) }
```

Let C_1 be the set of collections to which `Bombing-100` belongs[17]. In other words, C_1 is the set of all bindings for ?x in the query `(isa Bombing-100 ?x)`. Therefore, the estimate for the query would be given by:

$$max_{col \in C1} SuccessEstimate(eventPartiallyOccursAt, col, 1, 1) \quad \text{... (2)}$$

For simplicity, let us assume that C_1 is {Bombing}. Then the estimate of success for the query reduces to:

$$SuccessEstimate(eventPartiallyOccursAt, Bombing, 1, 1) \quad \text{…(3)}$$

The query can be answered in two ways: (i) by ground facts and (ii) by using axioms. The sum of estimates from these two sources will be our approximation for expression 3 shown above.

- **Proof by ground facts**: This case corresponds to the situation when the answer already exists in the KB. The following expression represents our estimate. Since *depth* is zero for ground fact retrieval, we can use Equation 1b to evaluate it.
  $SuccessEstimate(eventPartiallyOccursAt, Bombing, 1, 0)$ …(4)

[17] The selection of collections for this task is an interesting open question.

- **Proof by axioms:** Here, we are trying to answer an `eventPartiallyOccursAt` query. Therefore, both axioms A_1 and A_2 are relevant. In general, let Axioms(p) be the set of relevant axioms for answering a query involving p. Then, expression 3 reduces to:

$$SuccessEstimate(eventPartiallyOccursAt, Bombing, 1, 0) + \sum_{a \in Axioms(p)} (Estimate\ from\ axiom\ a)$$
…(5)

where, p ← eventPartiallyOccursAt

Axioms(p) ← {A_1, A_2}

The procedure for collecting estimates from search paths is shown in Figure 4.2. Let us consider the axiom A_1 first. In step 2 of the algorithm v is set to ?ATTACK.

Procedure: Collect-Estimate- From-Axiom (a, q, d, c)

Input: An axiom: *a*.

A query: *q*

Depth parameter: *d*

A collection: *c*

```
1.  estimate ← 1
2.  v ← Variable in the consequent of the axiom which is bound to the object (or the entity) in
    the query q.
3.  for all terms t in antecedents of the axiom a do:
4.        If variable v occurs in term t, then:
5.            Let Predicate(t) represent the predicate in term t.
6.            Let n ← Argument Position of variable v in term t.
7.            estimate ← estimate× $SuccessEstimate(Predicate(t), c, n, d - 1)$
8.        endif
9.  endfor
10. return estimate
```

Figure 4.2: Procedure for estimating the utility of axioms.

When t is `(isa ?ATTACK AttackOnObject)`, we calculate $SuccessEstimate(isa, Bombing, 1, 0)$ in step 7. Similarly, when t is `(objectAttacked ?ATTACK ?OBJECT)`, we calculate $SuccessEstimate(objectAttacked, Bombing, 1, 0)$.

Therefore, the complete definition of the *SuccessEstimate* function is given by the following expression:

$$\boldsymbol{SucessEstimate(p, c, pos, d)} \leftarrow \boldsymbol{SuccessEstimate(p, c, pos, 0)} + \mathbf{A} \qquad \textbf{where}$$

$$\mathbf{A} \leftarrow \left(\sum_{\boldsymbol{a} \in \boldsymbol{Axioms(p)}} \prod_{\boldsymbol{t} \in \boldsymbol{Antecedents(a)}} \boldsymbol{SuccessEstimate}(\boldsymbol{Predicate(t)}, \boldsymbol{c}, \text{argpos}(\text{t}, \text{a}, \text{pos}), \boldsymbol{d-1})\right)$$
$$\boldsymbol{when\ d} > 0$$
…(6)

The term[18] argpos(t, a, pos) represents *n* in step 6 of Figure 4.2. It returns the argument position of variable *v* in *t*.

Moreover, for queries like `(eventOccursAt Bombing-1 Island-100)`, where `Bombing-1` and `Island-100` are both constants, we define the likelihood as:

min {*SuccessEstimate*(`eventOccursAt, Bombing, 1,` *`depth`*),

SuccessEstimate(`eventOccursAt, Island, 2,` *`depth`*)}.
…(7)

[18] arg-pos(t, a, pos) returns 0 and SuccessEstimate(p, c, 0, d) is set to 1 when the variable doesn't occur in the term. See 3.i in Figure 2.

Expressions 5 and 6 overestimate the likelihood of proving the query for two reasons: (1) We make the simplifying assumption that clauses of an axiom are independent and (2) we assume that axioms are independent of each other. Addressing these assumptions is an important open question, but as we shall see, these estimates are still quite useful in pruning queries that are relatively unlikely. Importantly, *SuccessEstimate* can be computed off-line, via a static analysis of the KB contents. It can be efficiently

Algorithm: **PAS (goals, θ, α, d)**

Returns a set of answers to *goals*

Input:

- A list of conjuncts forming a query, *goals*
- Θ, the current substitution, initially the empty substitution {}.
- A threshold, α.
- Depth limit, d
- K: The set of axioms in the Knowledge base.

```
1. Let solutions ← Φ.
2. If goals is empty return {Θ}
3. q' ← SUBST (Θ, FIRST (goals)).
4. For each axiom r in K where r = $p_1(x) \wedge p_2(y) \wedge \ldots p_n(z) \rightarrow q(w)$ and Θ' ← Unify(q(w), q')
   succeeds, let s = predicate in q(w)
5. Let $v_1, v_2 \ldots v_n$ be the argument positions of variables in q(w)
6. Let U = {$U_1, U_2 \ldots U_n$ }be the set of most specific collections of the entity in argument
   position $v_1, v_2 \ldots v_n$ respectively.
7. $If\ min_{Ui \in U}\ max_{ci \in Ui}\ SuccessEstimate(s, c_i, v_i, d) > \alpha$
       i. goals' ← [$p_1(x), p_2(y), \ldots, p_n(z)$| REST(goals)]
       ii. solutions ← PAS (goals', COMPOSE(Θ', Θ), α, d-1, K) ∪ solutions
8. return solutions
```

Figure 4.3: Description of the algorithm *PAS*

implemented as a sparse lookup table, since the overwhelming majority of the possible combinations of arguments will yield a result of zero. Although we have not yet done so, this table would be straightforward to update as learning occurs, since all of the computations are relatively local. By pre-compiling *SuccessEstimate*, we can use it as an oracle during inference to quickly prune unproductive queries. The algorithm, ***PAS***(α) (PAS stands for 'Prune and Solve'), shown in Figure 3, is a standard backward chaining algorithm [Russell & Norvig 2003] with an extra constraint in step 7 which prunes sub-queries with likelihood below α. In section 4.5, our experimental evaluation shows that this heuristic helps provide significant memory savings.

4.4 An Empirical Approach for Estimating the Effectiveness of Axioms

The previous section described an analytical method for estimating the efficacy of axioms. This section describes an alternative, a method for learning, from a training set of queries, estimates of the probability that an axiom will be used successfully in a proof. This approach assumes that the training set and the test set have similar distribution of queries. The training set consists of a set of randomly chosen queries. The proofs of these queries are computed and used to estimate rule success probabilities, which are stored in a database and used to prune and/or prioritize search paths.

Our goal is to estimate the probability that a given axiom will be successful in a query. Recall that we are limiting inference to Horn axioms. Let A_1 A_2 …A_n be the antecedents of the axiom H and C be H's consequent. We define *AxiomSuccess*(H) to be an estimate of the probability that solutions will be found for C via axiom H given the current KB within a fixed time interval. To ensure consistency, the probabilities of all rules for a particular consequent must sum to 1. In other words:

$$\sum \mathit{AxiomSuccess}(r)=1 \text{ for } r\varepsilon A, \text{ where } A=\{\, r \mid \text{Consequent}(r) = C\} \quad \forall C \qquad \text{(Equation 8)}$$

We begin with a set of rules and a training set of queries. The EP algorithm for computing *AxiomSuccess* is shown in Figure 4.4. In Step 1 of the algorithm, we impose a uniform probability distribution on the rules by way of initialization. (The uniform distribution can be replaced by any consistent distribution.) In step 2, we calculate the probability of each proof tree. The probability of a proof tree is the product of all rules used in deriving the tree. In Step 2, $f_r(t)$ is the number of occurrences of the rule *r* in the tree *t* and p(r) is the probability estimate of the rule *r*. Some queries lead to more than one answer. Therefore, in Step 3 of the algorithm, we normalize the weight of all proof trees. The set S(q) in Step 3 represents all proof trees for the query *q*. The denominator in Step 3 sums the probability of this set and the relative weight of each tree in S(q) is computed. In Step 4, f(r) is the weighted frequency of the rule *r*. In Step 5,

Algorithm: EP

Notation:

R: the set of all axioms

T: the set of all proof trees in the training set.

Q: the set of all queries in training set.

S(q): All proof trees for the query q.

```
1. For all rules r ε R, set $p(r) \leftarrow \frac{1}{|N_a|}$ where $N_a \leftarrow$ {s | Consequent(s)= a}
2. For each t ε T, calculate $p(t) \leftarrow \prod p(r)^{fr(t)}$
3. For each q ε Q and t ε S(q), calculate $\boldsymbol{w(t)} \leftarrow \frac{\boldsymbol{p(t)}}{\sum_{\boldsymbol{u \in S(q)}} \boldsymbol{p(u)}}$
4. For each rule r ε R, set $\boldsymbol{f(r)} \leftarrow \sum_{\boldsymbol{Q(r)}} \boldsymbol{w(t)}$ where Q(r) is the set of all proof trees in
   which the rule r is used.
5. Set $AxiomSuccess(r) \leftarrow \frac{f(r)}{\sum_{A(r)} f(r)}$ where
   A(r) ⟵ { s | Consequence(s)= Consequence(r)}
```

Figure 4.4: Algorithm for estimating probabilities of rules.

the denominator sums the frequencies of all rules which have the same consequent as *r*. The numerator is the number of times this consequent is actually proved by using the expansion pattern represented by the antecedents of *r*.

We have found that most of the axioms have p(r)=0, i.e. we estimate that they will never be used. One possible explanation for this phenomenon is discussed in the next section. In Section 5, we prune all rules which had p(r)=0 and measure the performance. We show that such an approach is very helpful for reducing the resource requirements of deductive reasoning.

4.5 Global Semantic Analysis of Axioms

As AI systems use more axioms, the time required to reason with them increases far more rapidly than the increase in coverage. Sometimes the number of questions answered may even decrease! The reason is that, in any practical setting, time-outs are used to limit the amount of resources used in a query. Increasing the number of axioms increases the probability that the search process becomes lost in less useful regions of the search space. The techniques described so far provide two ways to guide search, the first by using static analysis to compile a table of likelihoods of success for particular queries, and the second by using empirical estimates of success, based on training data. Here we introduce a third technique, which builds on the static analysis described in Section 4.3 to estimate when queries are likely to succeed based on combining the constraints implied on types across antecedents.

As discussed above, we have found the antecedents for many axiom tend to unify successfully only for a very small number of collections, and that set of collections is more tightly constrained than the argument constraints of the predicates in the axiom. For example, consider the following axiom from the ResearchCyc KB:

```
(← (objectFoundInLocation ?COMMANDER ?LOC)
   (objectFoundInLocation ?UNIT ?LOC)
   (unitCommander ?UNIT ?COMMANDER))                                  (Axiom 2)
```

Axiom 2 is not useful for queries like `(objectFoundInLocation AlbertEinstein ?x)` because the second constraint is not likely to be satisfied. It is mainly relevant for inferring spatial locations of military-related entities. However, such a constraint is not explicitly part of the axiom. Partitioning might help, e.g., Axiom 2 might be stored in a microtheory that is designed for military reasoning. However, partitioning cannot solve this problem in general because commonsense problems typically involve knowledge from a broad range of domains. Now consider this axiom:

```
(<== (temporallyCoterminal ?BLO ?DYING)
     (subEvents ?KILLING ?DYING)
     (isa ?DYING Dying)
     (organismKilled ?KILLING ?BLO))                                  (Axiom 3)
```

Consider using this axiom to find the bindings for the variable ?DYING, given an entity for ?BLO. As it happens, the knowledge base has subEvents assertions about instances of collections like War, Hijacking, MilitaryConflictEvent and PsychologicalWarfare. None of these collections is a specialization of Dying. Therefore, we can conclude that the second and third constraints would never be satisfied if the depth limit is set to 1. Similarly, we expect instances of collections like ShootingSomeone, LynchingSomeone, Crucifying and Electrocution in the first argument position of organismKilled. This set is inconsistent with the types of information available for the subEvents predicate. Therefore, we can say that the first and last constraints of axiom 3 are unlikely to be mutually satisfied. Therefore, the distribution of facts in KB suggests that the solution space for this axiom is a null set and this axiom is not useful for answering queries.

These examples suggest that axioms can fail to produce results in at least two ways: (a) When an entity in the query is not included in the small space induced by one of the constraints of the axiom (e.g., see Axiom 2 and the query (objectFoundInLocation AlbertEinstein ?x)). (b) Secondly, the axiom can also fail to produce results when we cannot find any bindings for one of the internal variables in the constraints. (e.g., ?KILLING in Axiom 3).

In other words, the variables in the antecedents have their bias for instances of certain collections. There are several potential reasons for such biases. One source is that the world itself has constraints, e.g. academic conferences don't have maleficiaries. Another source is that any commonsense KB, no matter how large, is going to be incomplete. Depending on how it is constructed and extended, its distribution of ground facts may or may not be representative of the distribution of relationships and properties of entities in the real world. But in any case, deductive reasoning over the KB must take these biases into account. When these biases are combined across the antecedents of an axiom, the resulting solution space is much smaller than the solution space of any of the constraints in the antecedents. In other words, our ability to prove different constraints of the axioms is limited to instances of very dissimilar collections. Local search routines tend to adopt locally optimal configurations and find it difficult to merge them to form global solutions because the search frontiers of different terms in antecedents do not intersect. Similar problems have been identified in statistical physics and CSP community [Mackworth 1977, Mezard *et al* 2002, Braunstein *et al* 2005]. The basic idea of these algorithms is that we should collect information about these biases and guide the search towards values which are likely to be consistent with all constraints. As noted in Section 4.1, the space represented by set of values which are consistent with the preferences of all constraints is called the global constraint space. Our approach of finding useful axioms is based on the idea that axioms should use the global constraint space to prune less useful axioms. We now formalize this intuition, defining the idea of *likely collections*.

Let *a* be a Horn axiom, with *Antecedents*(a) and *Consequent(a)* representing the antecedents and consequent of *a* respectively. Let *t* ε *Antecedents*(a) be one of the constraints, with *Predicate*(t) and *Variables*(t) representing its predicate and variables.

Definition 4.7: We define *Collections*(t, pos, d) as:

$$Collections(t, pos, d) \leftarrow \{x | SuccessEstimate(\text{Predicate(t)}, \text{x}, \text{pos}, \text{d}) > 0\}$$

For example, let *a* be the Axiom 3 shown above and let *t* be the constraint `(organismKilled ?KILLING ?BLO)`. Then *Predicate*(t)= `organismKilled` and Variables(t) = {?KILLING, ?BLO}. Let us concentrate on the first argument position of `organismKilled`. Then `pos` is 1. From the definition shown above, we can infer that:

$$Collections(t, 1, d) \leftarrow \{x | SuccessEstimate(organismKilled, \text{x}, 1, \text{d}) > 0\} \qquad \ldots(9)$$

In the equation shown above, *Collections(t, pos, d)* represents the collections whose instances are likely to occur in the first argument position of `organismKilled` predicate when the depth of inference is limited to *d*.

Let us assume that equation 5 reduces to following expression:

$Collections(t, 1, d)$ ← {`ShootingSomeOne`, `LynchingSomeone`, `Crucifying`, `Electrocution`}

Collections(t, pos, d) is an indication of preferred collections for variable in argument position *pos* in *t*. We believe that an inference engine should prune those queries in which variables are not bound to instances of these collections. If *Collections(t, pos, d)* = Ø for a given *pos*, then we predict that the inference engine would not be able to find any binding for the variable in that position. The collections for which the axiom would return results are calculated by accumulating evidence from all constraints. Next, we define *LikelyCollections* for representing the preferences of all constraints in a given axiom.

LikelyCollections(*a*, *t*, *pos*, *d*) is the set of collections which are consistent with all constraints of the axiom, i.e., it represents the global constraint for the axiom. That is, the axiom *a* will lead to an answer only when the variable in position *pos* of term *t* is assigned a value which is an instance of collections in the aforementioned set. Figure 5 shows the algorithm for calculating *LikelyCollections*. Next we discuss how this method would work for the Axiom 3 shown above.

```
Procedure: FindLikelyCollections (a, t, pos, depth)

Input: An axiom: a
        A term: t
       An argument position: pos
       A depth parameter, depth

 1. Let $v \leftarrow$ Variable in the argument position pos of t.
 2. Let $p \leftarrow$ Predicate in term t
 3. Let $LikelyCollections\ (a, t, pos, depth) \leftarrow \boldsymbol{Domain}^c(p, pos)$.
 4. for all terms $q$ in the antecedents of axiom $a$, do:
 5.       if variable $v$ occurs in term $q$ then:
 6.             Let $p' \leftarrow$ Predicate in term $q$
 7.             Let $pos' \leftarrow$ Position of variable $v$ in $q$.
 8.             $LikelyCollections\ (a, t, pos, depth) \leftarrow LikelyCollections(a, t, pos, depth) \cap$
                                                   $Collections(q, pos', depth - 1)$
 9.       endif
10. endfor
11. return $LikelyCollections\ (a, t, pos, depth)$.
```

Figure 4.5: Procedure for calculating likely collections

Let *a* be the axiom 3 shown above. Moreover, let *t* represent the term (subEvents ?KILLING ?DYING).In step 3 of Figure 4.5, *LikelyCollections*(a, t, 1, d) is initialized to all specializations of Event[19]. The execution of steps 4-10 is shown below:

$$LikelyCollections(a,t,1,d)$$
$$\leftarrow Collections\big((organismKilled\ ?KILLING\ \ ?BLO),1,d-1\big)$$
$$\cap\, Collections\ ((subEvents\ ?KILLING\ ?DYING),1,d-1)$$

$\xrightarrow{yields}$

$\{x|SuccessEstimate\ (orgamismKilled,x,1,\ \ d-1)>0\}\ \cap$
$\{y|SuccessEstimate(subEvents,y,1,\ \ d-1)>0\}$

$\xrightarrow{yields}$ {ShootingSomeone, LynchingSomeone, Crucifying, Electrocution} ∩

{War, Hijacking, MilitaryConflictEvent, PsychologicalWarfare}

$\xrightarrow{yields}$ Φ

LikelyCollections(a, t, pos d) represents the collections which are consistent with the biases of all constraints of axiom *a* when the depth of inference is limited to *d*. Let *DomainOfVariable (Consequent(a), pos)* represent the domain of variable in the argument position *pos* of the consequent of axiom *a*. Then the denominator of the expression shown below represents the total number of instances

[19] `Event` is the argument constraint for the first argument of `subEvents`.

which satisfy the argument constraints. A typical theorem prover would use the axiom for all these instances. On the other hand, the numerator[20] is the total number of instances which belong to the collections in *LikelyCollections*(a, t, pos, 3). The difference in the size of these sets provides insight about the utility of the axiom. We plot γ(a, 1) and γ(a, 2) in Figure 4.6.

Definition 4.9:

$$\gamma\ (i, pos) = \frac{|\ \bigcup_{collection\ \in\ LikelyCollections\ (\ i, Consequent(i), pos, 3)}\{ins | ins\ \in collection\}\ |}{|\ DomainOfVariable(Consequent(i), pos)|}$$

When γ(i, pos) is small, the theorem prover uses the axiom for many queries, but most of them do not work out. On the other hand, when γ(i, pos) is close to 1, the axiom is likely to be useful for most queries which satisfy the argument constraint.

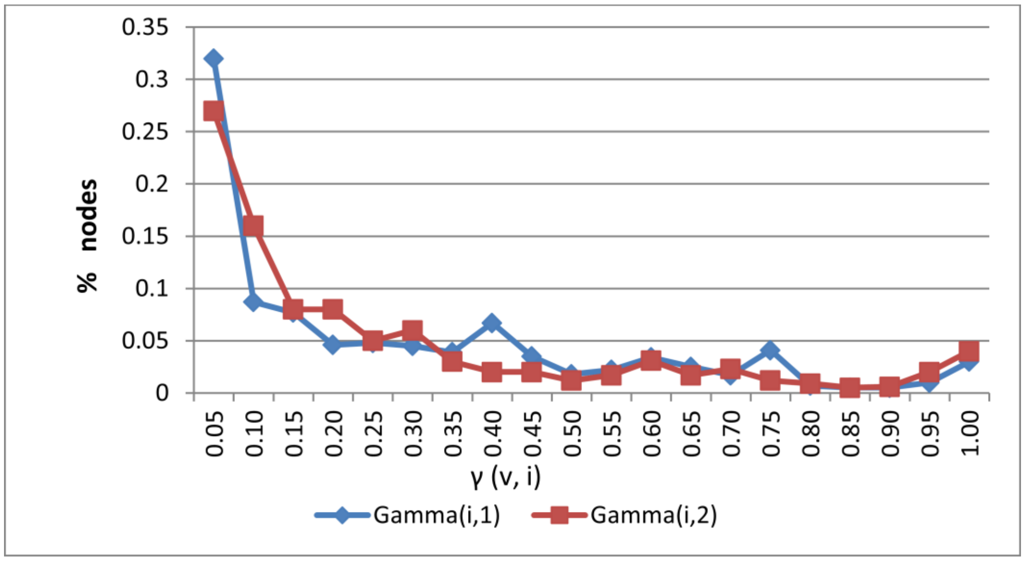

Figure 4.6: Distribution of γ(i, pos).

In Figure 4.6, we observe that for about 33% nodes, γ(a, 1) is 0. In other words, these axioms are not useful for instances of any collections. Moreover, most nodes have very low values of γ(a, pos). Less than

[20] In this work, the depth of inference is limited to 3. Therefore, without loss of generality, we study the properties of LikelyCollections(a, t, pos, 3).

10% nodes have values between 0.1 and 0.15. This means that |*LikelyCollections*(a, t, pos,d)| is very small for a significant number of axioms. We use this observation to prune such axioms. The algorithm **AxiomSelect**(β), shown in Figure 4.7, uses this idea to prune axioms. We have augmented the standard backward chaining algorithm [Russell & Norvig 2003], i.e., in steps 5 and 6, we prune all axioms which have γ(i, j) values below a threshold.

```
Algorithm: AxiomSelect (goals, θ, β)

Returns: a set of answers to goals

Input:

•  A list of conjuncts forming a query, goals
•  Θ, the current substitution, initially the empty substitution {}.
•  A threshold, β.


1. Let solutions ← Φ.
2. If goals is empty return {Θ}
3. q' ← SUBST (Θ, FIRST (goals)).
4. K ← Set of axioms for proving q'
5. RejectedAxioms ← $\{i \in K \mid \gamma(i, 1) \leq \beta$ or $\gamma(i, 2) \leq \beta\}$, where $v(i, j)$ is the jth variable in
   the consequent of axiom i.
6. K' ← K – RejectedAxioms
7.
8. For each axiom r ε K' where r = $p_1(x) \wedge p_2(y) \wedge \ldots p_n(z) \rightarrow q(w)$ and Θ' → Unify(q(w), q')
   succeeds
        a. goals' ← [$p_1(x), p_2(y), \ldots, p_n(z)$| REST(goals)]
        b. solutions ← AxiomSelect (goals', COMPOSE(Θ', Θ), β) U solutions
9. return solutions
```

Figure 4.7: Description of AxiomSelect (β)

4.6 Similarities with Arc Consistency Methods

The method described above explains how the domain of queries can be filtered. In what follows, we explain the similarities between arc-consistency algorithms in the CSP literature [Mackworth 1977] and the domain filtering heuristics described here. There is another way in which this problem can be observed. Note that axiom 4 and 5 shown below are equivalent:

```
(<== (temporallyCoterminal ?BLO ?DYING)
     (subEvents ?KILLING ?DYING)
     (isa ?DYING Dying)
     (organismKilled ?KILLING ?BLO))                       …(Axiom 4)
```

```
(<== (temporallyCoterminal ?BLO ?DYING)
     (subEvents ?X ?DYING)
     (isa ?DYING Dying)
     (organismKilled ?Y ?BLO)
     (equals ?X ?Y))                                        …(Axiom 5)
```

The domains for variables ?X and ?Y are given by the instances of `Event` and ActionOnObject respectively. As discussed above, these domains can also be represented using the specializations of these collections. For example, the domain for ?X is given by:

Domain^{c}(subEvents, 1) = {`EconomicEvent, DiscoveryEvent, (CausingFn Rioting), DangerousEvent, …`}

Similarly, the domain for ?Y is given by:

Domain^{c}(organismKilled, 1) = {`BlowingSomethingUp, CuttingSomething, Vandalism, Hijacking, KillingByOrganism, …`}

Then the algorithm shown below combines the preferences of variables ?X and ?Y and creates a list which would be a superset of the possible bindings for the variable ?KILLING in Axiom 4.

Procedure Revise Domains (x, y)

Input: Two variables x, y and their domains D_x and D_y.

Output: D_x, such that x is consistent relative to y.

```
1. for each a ε $D_x$
2.     if there is no b ε $D_y$ such that a=b
3.         then delete a from $D_x$
4.     endif
5. endfor
```

Figure 4.8: The Revise Procedure

The procedure shown in Figure 4.8 should be executed be all pairs of variables. It is equivalent to AC-1 arc consistency algorithm discussed in [Mackworth 1977, Dechter 2003]. In CSP problems, step 2 is written in a more general way as shown below:

$$\boldsymbol{if\ there\ is\ no\ b \in D_y\ such\ that\ (a,b)\ \in\ R_{xy}}$$

The unification needed in first-order axioms leads to the equals condition in Axiom 5. Therefore, the condition shown above has been simplified to step 2 in Figure 4.8.

4.7 Experimental Method and Results

To check the robustness of our methods, we sampled queries from knowledge rich sections of a KB. We used 1.2 million facts from the 2007 version of ResearchCyc KB, including everything but the provenance information available in that build[21]. Let *KnownFacts*(p) represent the number of ground facts about the predicate *p*. We define *InferredFacts*(p) as the sum of ground facts of all nodes below *p* in the genlPreds lattice. Formally it is $\sum_X$ *KnownFacts*(x), where X is the set of predicates reachable from p via genlPreds links pointed downward. We define *AllFacts(p)* as *KnownFacts(p)* + *InferredFacts(p)*, i.e., the total number of times a predicate *p* could be proved via ground facts and genlPreds inference. We sorted all predicates by *AllFacts(p)* and then selected top 55 predicates from these sets. To get a query set, we replaced the first argument of these predicates by 20 randomly sampled entities satisfying their argument constraints to get a set of 1100 queries. A variable was used for the second argument. The statistics of the knowledge base are such that all of these queries were binary predicates, hence each query had only one open variable.

We repeated this process 5 times to get 5 sets of queries. The algorithms proposed in this paper, ***PAS***(α)**,** **EP** and ***AxiomSelect***(β), have been tested on these 5 sets each containing 1100 queries. For a baseline comparison, we included all pure Horn axioms for these 55 predicates and their subgoals through depth 3. This set had 6500 axioms. The axioms in these experiments were not range restricted [Manthney & Bry 1988]. We used a backward-chaining algorithm [Russell & Norvig 2003] implemented in an LTMS-based

[21] The provenance information includes when a fact was entered, by whom, and other information useful in knowledge entry quality control.

inference engine. [Forbus & de Kleer 1993] The simple backchaining algorithm was augmented by simple heuristics like constraint reordering [Bekes & Szeredi 2007]. We sought all answers for the queries in a set, with a depth limit of three. Each query was timed out after 20 minutes. All experiments were done on a 3.2 GHz Pentium Xeon processor with 3GB of RAM.

Our results are summarized in Tables 4.1 and 4.2. We have experimented with a range of values of α for these algorithms. Our best results for *PAS* and *AxiomSelect* were obtained for $\alpha = 10^{-4}$ and $\beta = 0.3$ respectively. On an average, ***PAS***(α***)*** leads to 2.8% loss in completeness with a factor of 4.8 speedup and 86% fewer node expansions. Similarly, on average, ***AxiomSelect***(β) leads to 4.38% gain in completeness with a factor of 16.5 speedup and 98% fewer node expansions. For the **EP** algorithm, we used a normal backward-chaining algorithm and removed all rules with p(r)=0. It leads to 10.16% gain in completeness with a factor of 135.35 speedup and 98.48% fewer node expansions.

We have studied how these algorithms depend on their parameters. Figure 6 and 7 show the performance of *PAS(α).* Time, space and coverage of *PAS(α)* decrease steadily with α. However, memory requirements decrease at a faster rate than other two performance indicators.

Query Set #	Rule sets	% Answered	Time (minutes)	Nodes Expanded	%Improvement Over baseline	Speedup Over baseline	%Memory used w.r.t. baseline
1	Baseline	37.36	5445.22	13280753	-	-	-
	PAS(10^{-4})	40.27	1198.76	1901051	7.78	4.54	14.31
	EP	42.18	31.24	313920	12.90	174.30	2.36
	AxiomSelect(0.3)	41.00	338.79	185596	9.74	16.07	1.39
2	Baseline	43.18	4377.71	10157326	-	-	-
	PAS(10^{-4})	38.45	944.19	1305876	-10.95	4.63	12.85
	EP	43.90	37.67	378734	1.66	116.21	3.72
	AxiomSelect(0.3)	41.27	309.72	178116	-4.42	14.13	1.75
3	Baseline	38.81	5078.99	10712619	-	-	-
	PAS(10^{-4})	36.72	963.12	1811501	-5.38	5.27	16.90
	EP	42.90	35.89	361959	10.53	141.51	3.37
	AxiomSelect(0.3)	40.27	290.48	169402	3.76	17.48	1.58
4	Baseline	37.54	4634.96	10868485	-	-	-
	PAS(10^{-4})	35.27	911.07	1702866	-6.04	5.08	15.66
	EP	39.27	35.06	349350	4.60	132.20	3.21
	AxiomSelect(0.3)	36.36	284.58	179810	-3.14	16.28	1.65
5	Baseline	33.54	4521.03	12816567	-	-	-
	PAS(10^{-4})	33.72	1006.85	1179939	0.53	4.49	9.20
	EP	40.63	39.82	395786	21.13	113.53	3.08
	AxiomSelect(0.3)	38.90	243.11	163650	15.98	18.59	1.27

Table 4.1: Summary of Results

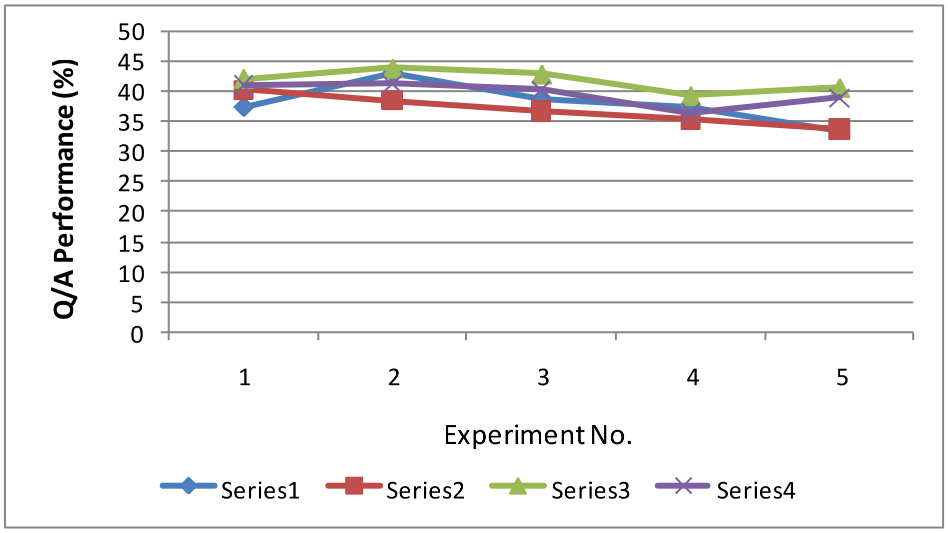

Figure 4.81: Performance of different algorithms shown in Table 4.1

On the other hand, there has not been much change in completeness in our experiments ($0.3 \leq \beta \leq 0.8$) for *AxiomSelect*. The maximum change in completeness was 1.6%, 1.36%, 1.7%, 1% and 1.4% for query sets 1-5 respectively. However, other performance indicators changed sharply in this range. Figure 8 and 9 show the trends in memory usage and time for *AxiomSelect*. We observe that memory usage increases at a much faster rate than time requirements.

Our results clearly indicate that these techniques lead to significant speedups with minimal loss in completeness. The databases are complied offline. The one-time cost of constructing these databases is O(|P|* |C|), where P is the set of predicates and C is the set of collections.

Algorithm	**Average % improvement in completeness over baseline**	**Average speedup over baseline**	**Average % memory used w.r.t. baseline**
PAS(α)	-2.81	4.80	13.78
EP	10.16	135.35	3.14
AxiomSelect(β)	4.38	16.51	1.52

Table 4.2: Average performance of algorithms

The three algorithms mentioned here aim at removing less useful regions of the search space. However, they use different strategies and insights for achieving this aim. The **EP** algorithm is the simplest among the three. From Table 1, it is clear that EP is most effective in improving the time requirements. However, PAS(α) and AxiomSelect(β) provide more insight into the reasons for poor Q/A performance of deductive reasoning engines. In particular, metrics like ***SuccessEstimate*** *(predicate, collection, argument-position, depth)* and *LikelyCollections*(variable, axiom, depth) provide important information about knowledge gaps in the KB. For example, if ***SuccessEstimate*** *(causes-EventEvent, Typhoid-Fever,* 1, 3*)* is equal to 0, it points toward lack of biological knowledge about Typhoid-Fever in our KB. Similar information can also be obtained from the *LikelyCollections* metric. For example, let A be the set of all axioms which have the predicate causes-EventEvent in the consequent. Then the following expression represents all collections whose causal consequences can be found. Here the function *var* (a, 1) returns the variable in the first argument position of the consequent.

$$Q = \bigcup_{a \in A} LikelyCollections(a, Consequent(a), 1, d)$$

Therefore, if Q does not contain Typhoid-Fever, then obtaining information about the causal consequences of Typhoid-Fever should be a reasonable learning goal. The algorithm *PAS* and *AxiomSelect* are based on similar ideas. From Table 2, it is clear that ***AxiomSelect***(β) is better than *PAS(α)* in all respects. This means that the global analysis of axioms and combination of biases of different constraints in the antecedents is very important. However, an empirical method like *EP* can provide significantly better performance than them.

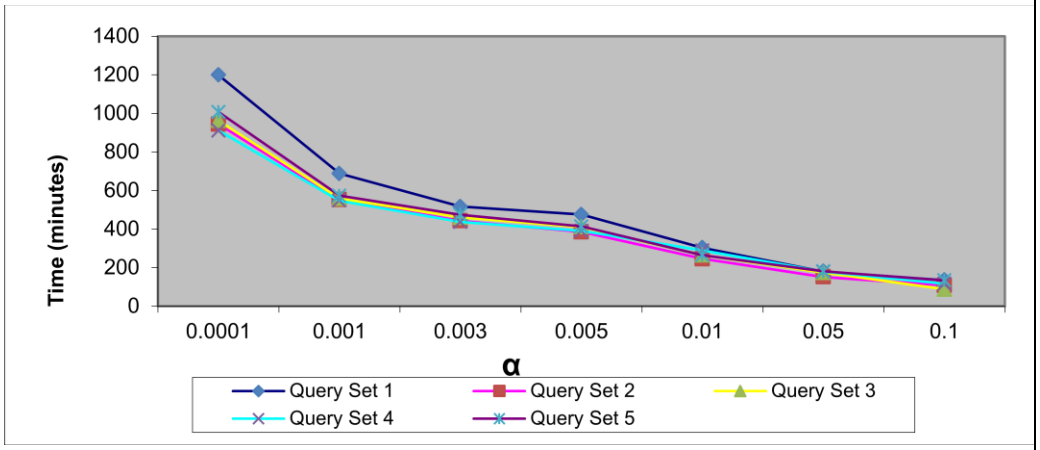

Figure 4.9: Scaling of PAS(α) with α.

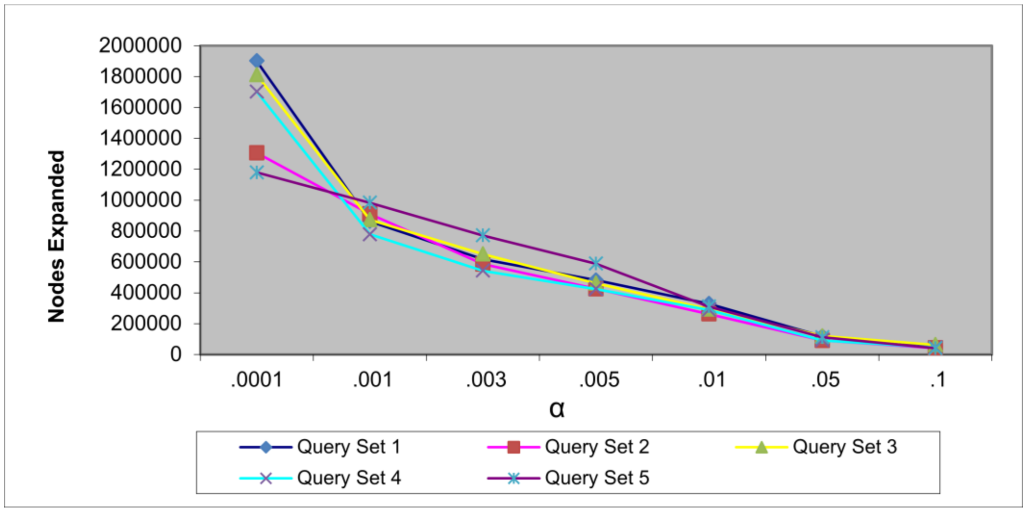

Figure 4.10: Scaling of PAS(α) with α.

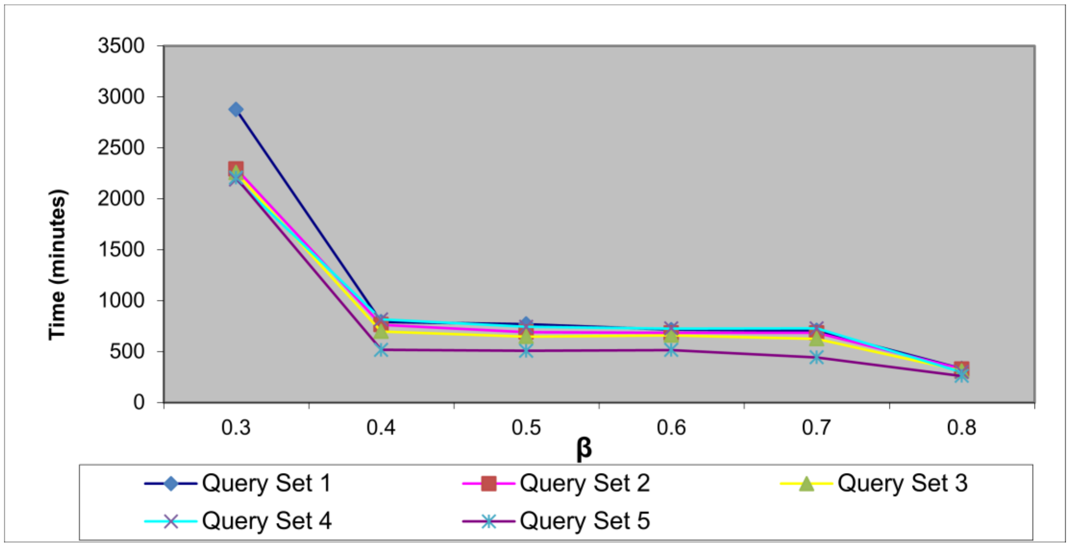

Figure 4.11: Scaling of AxiomSelect with β.

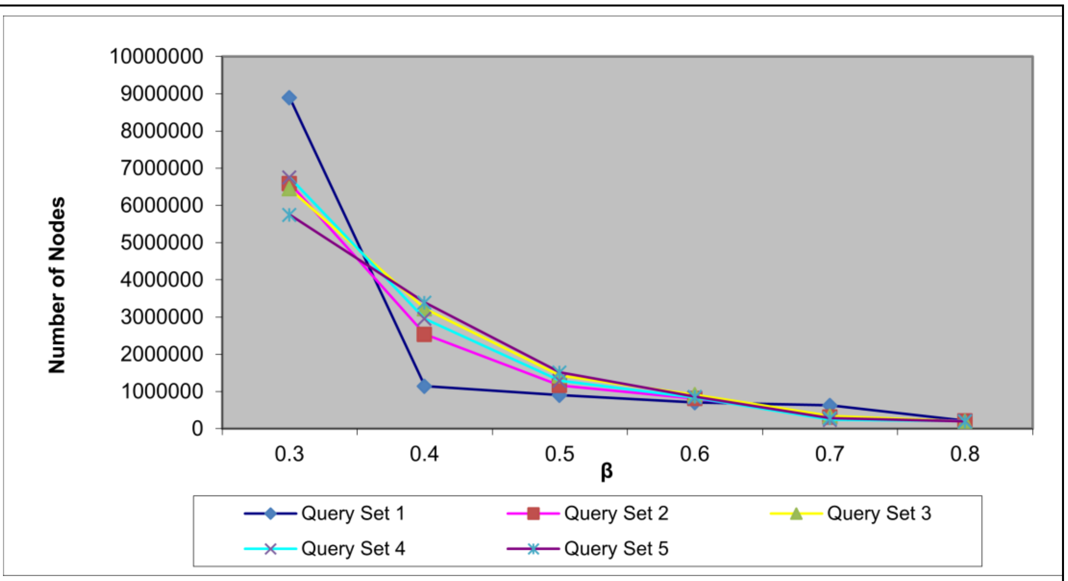

Figure 4.12: Scaling of AxiomSelect with β.

4.8 Related Work

The problem of extracting focused subsets of a knowledge base has been addressed before [Peterson *et al* 1998]. However, such work has generally focused on understanding syntactic relevance. By contrast, our work exploits the distribution of available knowledge in the KB and prunes branches which are unlikely to yield results. Research in computational complexity and knowledge compilation [Selman & Kautz 1996] has shown that Horn clauses provide a reasonable trade-off between expressiveness and tractability. However, as our experiments show, inference with Horn clauses isn't easy. We begin with a set of first-order Horn axioms and consider the problem of identifying a useful and tractable subset. Answer set programs are frequently compiled to propositional logic so that SAT solvers can be used [Baral 2003]. We do not consider the problem of optimization of propositional reasoning, which has been extensively addressed by the SAT reasoning community [Kautz & Selman 2007], because we are concerned with more general reasoning. Similarly, while the results of [Ramachandran & Amir 2005] on converting reasoning problems in monadic first order logic into equivalent problems in propositional logic may be theoretically interesting, most realistic knowledge bases involve relations with more than one argument.

[Lang *et al* 2003] discuss syntactic and semantic forms of independence and their role in improving inference. This approach requires deciding *a priori* whether all the axioms that might enter into the proof belong to a syntactic class. This seems difficult since problem instances involve different levels of prior knowledge. Researchers have addressed the problem of automatic finding of subgoal orderings [Ledeniov & Markovitch 1998]. The work presented in this paper complements such an approach because subgoal ordering should be performed after pruning less useful regions of search space.

In [Debray & Lin 1993], the authors have suggested a general algorithm for finding the number of solutions for logic programs. In [Greiner & Orponen 1991], the authors have proposed an algorithm that uses samples to approximate the probabilities needed to find near-optimal derivation strategies. Similarly, in [Frisch 1987] the author has studied some specialized inference schemes for knowledge-based systems. While these works are very important, they have not discussed how the distribution of facts in the external knowledge base can lead to significant unification-related problems. Moreover, we have shown that our methods work in a realistic AI system.

The idea of having an information gathering phase for understanding the dynamics of search has been explored in CSP literature [Wallace & Grimes 2008]. The notion of using a score which could be considered as a distance from a truth assignment to a solution has been explored in propositional logic literature [Sebastiani 1994]. Our work has primarily explored static analysis for optimization because any on-line decision making is likely to be very expensive, due to the size of search space. We have aimed at identifying useful regions of search space to prevent unproductive instantiation. The work on learning the probabilities of proof trees is based on learning probabilistic context free grammars [Prescher 2005].

Off-line compilation for fast run-time decisions has been studied by many researchers in fields like decision theory [Horvitz 1990], metareasoning [Russell & Wefald 1991], and game playing [Schaeffer *et al* 2005]. Such work has not addressed deductive reasoning. Existing work on deductive reasoning [Choi 1993] does not discuss the role of the distribution of facts in a KB. Our work is most similar to the work in survey propagation algorithms in statistical physics and CSP community [Mezard *et al* 2002]. To the best of our knowledge, there has not been any work in the deductive reasoning community which tackles the problem we address.

4.9 Conclusion & Future Work

Automatic optimization of axioms for deductive reasoning is an important problem for making AI systems that scale to use large knowledge bases. The importance of such techniques will grow as more large KBs are created, using advanced knowledge capture techniques such as learning by reading, including reading the web. The approach of semantic static analysis proposed here exploits information about the distribution of facts in the KB to ascertain when particular queries and axioms are unlikely to succeed. As our experiments show, the algorithms it supports can provide an order of magnitude speedup, with only a small loss in completeness.

Three lines of future work suggest themselves. As mentioned in the previous chapter, carrying out similar experiments on other large KBs would be informative. At this writing, we believe that they do not exist. While there are a reasonable number of medium to large scale ontologies in existence, the only knowledge about the terms in them tends to be structural knowledge and ground facts, not general axioms relating the terms in ways that support deep inference. Given the current interest in learning by reading and reading the web, we expect this to change, and we hope that the existence of more efficient reasoning techniques like these will help motivate the construction of more large-scale KBs. Second, our longer-term goal is to automatically identify knowledge that a learning system should seek, based on identifying gaps in its reasoning abilities. As suggested in Section 5, we believe that these techniques of semantic static analysis can be useful for this, since the distribution of incoming queries and their success/failure rates can be combined with both information about what is currently in the KB, and what might be learned, to suggest new learning goals. By identifying the kinds of new axioms and ground facts that it should be seeking, future learning systems would be able to improve their own performance over time. Finally, as KBs grow in size (given a fixed ontology), the density of facts would increase and pruning axioms/queries would not be possible. In such a situation, reasoning systems would need to order different regions of search space and traverse some of them given the time constraints.

5. Graph-Based Reasoning and Reinforcement Learning for Improving Q/A Performance in Large Knowledge-Based Systems

5.1 Introduction and Motivation

Knowledge-based systems typically use deductive reasoning for answering queries. However, the set of axioms available in KBs is inadequate for most applications. Due to these knowledge gaps, many questions remain unanswered and Q/A performance is affected. Since we do not expect to have a complete set of axioms in foreseeable future, we must find alternatives to rule-based deductive reasoning. We believe that plausible reasoning can play an important role in solving this problem. Here we argue that plausible reasoning can be seen as path finding in knowledge-bases. We also show that reinforcement learning can be used to learn the quality of these plausible inference chains.

In this chapter, we show how to integrate graph search, higher-order knowledge representation, and reinforcement learning to learn reliable patterns of plausible reasoning from ground facts. Given a fully ground query, we show how to incrementally search the facts which mention the entities in the query guided by a set of *plausible inference patterns* (PIPs). PIPs are expressed in terms of higher-order concepts in the knowledge base, specifically predicate type information. Since the number of predicate types is much smaller than the number of predicates, this greatly reduces the size of search space. We show that the quality of inference chains of PIPs can be learned by reinforcement learning.

We begin by discussing other related work. We then discuss the idea of PIPs and how they are defined in terms of predicate types. How reinforcement learning is used to learn the quality of answers is discussed next. We describe results and conclude in the final section.

5.2 Related Work

Researchers from the fields of Information Retrieval, Natural Language Processing, Databases and Logical Inference have contributed to the advancement of question answering technologies [Brill et al 2002, Prager et al 2004]. Overviews of question answering techniques can be found in [Belduccinni et al 2008, Molla and Vicedo 2007]. A comparison of challenge problems and different approaches has been discussed in a recent IBM report [Ferrucci et al 2009]. Learning Bayesian networks for WordNet relations for QA systems [Ravichandran and Hovy 2002] and surface patterns from natural language text [Molla 2006, Grois and Wilkins 2005] have been discussed. Our work is different in that we are trying to improve the performance of a plausible inference based Q/A system by learning to reason. Other learning to reason frameworks [Khardon 1999] have been explored. However, their efficacy for improving Q/A performance is not known. Reinforcement learning has been used for learning control rules for guiding inference in ResearchCyc KB [Taylor *et al* 2007]. To the best of our knowledge, there has not been prior work which develops a method for providing plausible explanations for queries (without using logically quantified axioms) with a learning framework.

5.3 Approach

The task of answering questions without using logically quantified axioms is difficult because it involves finding arbitrary relations between predicates which could explain the query. Therefore, we have taken the simpler approach of building a small sub-graph of relations around the entities in the query and then assessing the quality of inference chains between them. This intuition is similar to connection graphs [Faloutsos *et al* 2004] and relational pathfinding, where the domain is viewed as a (possibly infinite) graph of constants linked by the relations which hold between the constants [Richards & Mooney 1992]. Since prior knowledge is important for biasing learning, we leverage existing axioms in the KB to create *plausible inference patterns* (PIPs) which are used to keep only more likely inference chains. These PIPs are created by replacing predicates in axioms by their predicate types. PIPs are accepted if they are generated by more than N axioms. (In this work, N=5). We turn to a concrete example for illustration.

Let us assume that the system has been asked to provide a plausible inference for the query `(acquaintedWith BillClinton HillaryClinton)`. A small section of the KB relevant for answering this query is shown in Figure 5.1. In the first step of the algorithm shown in Figure 5.3, *e1* is set to `BillClinton` and *e2* is set to `HillaryClinton`. For simplicity, let us assume that we have just one PIP:

`FamilyRelationSlot(?x,?y) AND FamilyRelationSlot(?y,?z)` →

`PersonalAssociationPredicate(?x,?z)` **[PIP1]**

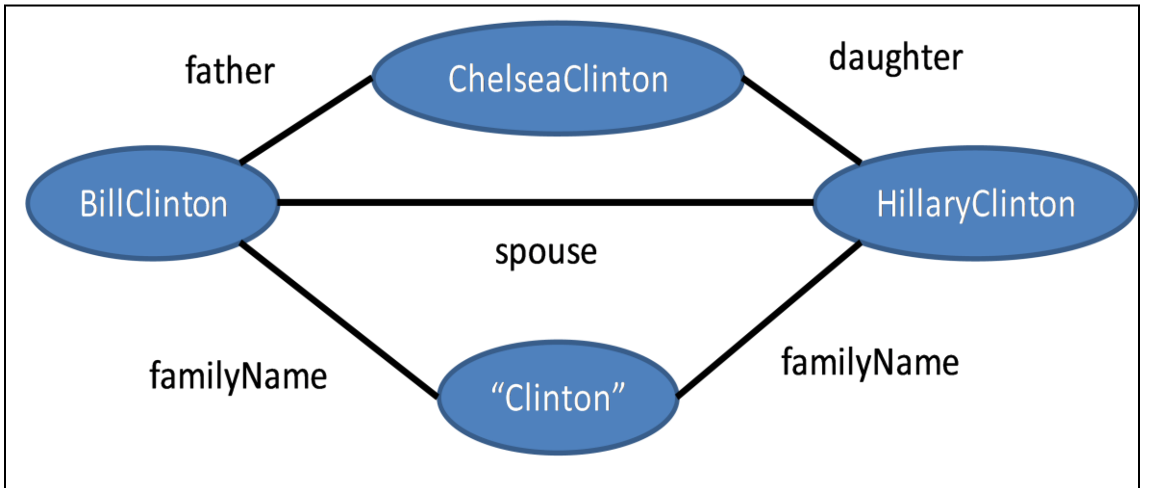

Figure 5.1: Plausible inference example

This pattern represents the knowledge that two predicates of type `FamilyRelationSlot` can plausibly combine to infer assertions involving personal associations. This representation has been chosen because we believe that predicate types like `SubEventPredicate`, `PhysicalPartPredicate` and `CausaltyPredicate` provide a meaningful level of abstraction for identifying plausible inference patterns. For instance, all predicates of type `SubEventPredicate` can be used for proving

Algorithm: **FindPlausibleExplanationsForQuery (FPEQ)**

Input: *query*: A query for which plausible explanations have to be found

Output: A set of facts which would justify *query*.

```
1. Let pred← predicate in query, e1← Entity in first argument position of query, e2← Entity
   in second argument position of query, Paths← Ø. Let Solutions← Ø.
2. Let patterns ← Relevant plausible inference patterns for pred
3. For each pattern in patterns
      a. Create a new path p.   Set p.completed ← Ø,  p.remaining ← Antecedents of
         pattern,  p.starting-entity ← e1,  p.target-entity ← e2, p.current-entity ← e1,
         Add p to the list Paths
4. For each p in Paths
      a. Let facts← All ground facts involving p.current-entity
      b. For each fact in facts
            1. Let ptypes ← Predicate types of the predicate in fact
            2. Let E ← Entities in fact
            3. For each constraint c in p.remaining
                  a. If c ε ptypes
                        i. Create a new path p1.   Initialize p1← p. Add
                           p1 to Paths
                       ii. p1.completed ← p.completed  +  c
                      iii. p1.remaining ← p.remaining – c
                       iv. p1.current-entity← E - p.current-entity
                        v. If p1.remaining = Ø and p1.current-entity =
                           p1.taget-entity then add p1 to Solutions and
                           Remove p1 from Paths
      c. Remove p from Paths.
5. Return Solutions
```

Figure 5.2

`eventPartiallyOccursAt` queries [22]. Similarly, all predicates of type `PhysicalPartPredicate` are relevant for proving `objectFoundInLocation` queries[23].

Therefore, learning knowledge in terms of predicate types is easier and more natural. The relative tractability of this formulation can also be seen by noting the difference between the sizes of search spaces. Learning to distinguish between correct and incorrect derivations of length k involves searching in a space of size N^k, where N is the size of vocabulary. In our KB, the number of predicates is 24 times larger than the number of predicate types. Therefore, learning PIPs in terms of predicate types is significantly easier. The algorithm FPEQ (see Figure 5.2) constructs a graph around the entities mentioned in the query and returns explanations which plausibly entail the query. Steps 1-3 perform initialization, with the path search being handled in Step 4. It maintains a list of partial paths which are extended by retrieving facts involving the frontier entities[24]. The algorithm terminates when all paths which match the antecedents of the available PIPs are found. In the previous example, `acquaintedWith` is a `PersonalAssociationPredicate` predicate; therefore the pattern PIP1 is relevant for this query. The algorithm shown in Figure 5.2 finds paths by successively expanding nodes at the frontier and keeps the partial paths in the list *Paths.* In step 3 of the algorithm, a new path *p* is created. Here, *p.starting-entity, p.target-entity* and *p.remaining* are set to `BillClinton`, `HillaryClinton` and

```
[FamilyRelationSlot(?x,?y), FamilyRelationSlot(?y,?z)]
```

[22] Some examples of `SubEventPredicate` predicates are `firstSubEvents`, `cotemporalSubEvents`, `finalSubEvents` etc.

[23] Some examples of `PhysicalPartPredicate` are `physicalParts`, `internalParts`, `northernRegion` etc.

[24] The algorithm shown in Figure 3 has been simplified for clarity. In particular, the algorithm keeps track of bindings of variables and paths can only be extended when bindings are consistent.

respectively. Essentially, this means that we are looking for a path between the nodes labeled

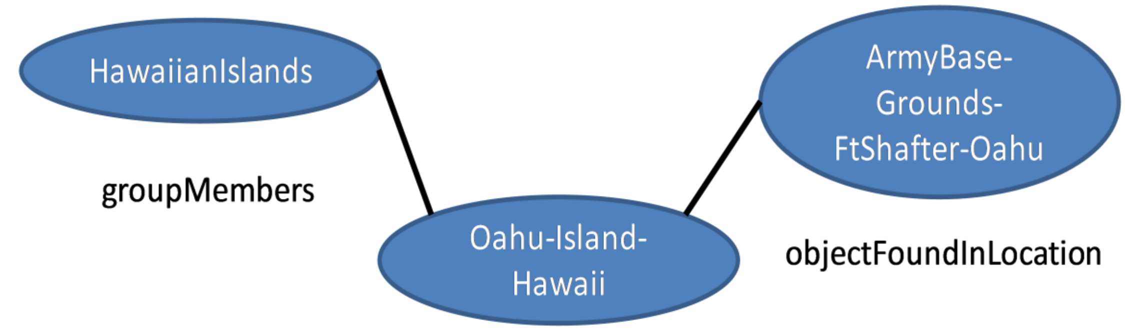

Figure 5.3: Another plausible inference example

`BillClinton` and `HillaryClinton` traversing two edges labeled with predicates of type `FamilyRelationSlot`. In Figure 5.1, a small section of the graph is shown[25]. In step 4 of the algorithm, all facts involving the current entity (`BillClinton` in this example) are retrieved. The partial path *p* is extended by including the fact `(father ChelseaClinton BillClinton)` in the partial proof. At this stage, we are looking for a path from the node `ChelseaClinton` to `HillaryClinton` such that the edge label is a predicate of type `FamilyRelationSlot`. Another expansion of this path with the fact `(mother ChelseaClinton HillaryClinton)` satisfies this requirement, and the path is added to the solutions[26]. The second path involving two edges labeled '`familyName`' would not be selected because no PIPs use predicates of type `ProperNamePredicate-Strict` to entail `PersonalAssociationPredicate` predicates. Similarly, the PIP shown below would help in proving `(objectFoundInLocation ArmyBase-Grounds-FtShafter-Oahu HawaiianIslands)` (see Figure 5.3)[27].

```
SpatialPredicate(?x, ?y) Group-Topic(?z,?y) →
                                         SpatialPredicate(?x, ?z)
```
… **[PIP2]**

[25] For simplicity, the direction of arcs is not shown. The order of arguments is represented in the PIPs.

[26] The algorithm shown in Figure 3 only finds simple proofs. A more complete proof procedure would involve finding a spanning tree between the entities. This can be done by ensuring that *p.current-entity* is a list of entities (and not a single entity).

[27] We note that `groupMembers` and `objectFoundInLocation` are instances of `Group-Topic` and `SpatialPredicate` respectively.

The FPEQ algorithm uses the predicate type hierarchy. Our inference scheme also simplifies inference by condensing inference chains. For example, `wife` is a `PersonalAssociationPredicate`; therefore the inference from wife to `acquaintedWith` is a one-step process. On the other hand, using the normal predicate type hierarchy involves multi-step inferences. For example, the inference chain from `wife` to `acquaintedWith` requires following axioms.

```
(← (acquaintedWith ?x ?y)
   (mutualAcquaintances ?x ?y))
(← (mutualAcquaintances ?x ?y) (mate ?x ?y))
(← (mate ?x ?y)(spouse ?x ?y))
(← (spouse ?x ?y) (wife ?x ?y))
```

As discussed above, the number of predicate types is less than the number of predicates. Therefore, the predicate type hierarchy maps the predicates to a smaller space. This phenomenon speeds up the search because the average path length between two nodes in this smaller space is less than what we encounter in a typical predicate hierarchy. This plays an important role in improving inference in resource constrained Q/A systems.

The FPEQ algorithm can be easily extended to handle queries with variables. This would entail checking that the node at the search frontier satisfies the argument constraint of the predicate. For example, let us consider the query `(acquaintedWith BillClinton ?x)`. When the search process reaches the node labeled `HillaryClinton`, it would notice that all antecedents of the PIP have appropriate bindings and the entity `HillaryClinton` satisfies the argument constraint of the second argument position of `acquaintedWith`. Such inference chains will be included in the solutions.

5.4 Learning to Reason

Many learning systems find the correct level of generalization by trial-and-error. Our approach gets initial plausible inference patterns by replacing predicates in axioms by their predicate types. These generalizations certainly increase the deductive closure but can lead to erroneous answers. For example, the pattern PIP2 shown above would lead to an incorrect answer if we use `bordersOn` as an instance of `SpatialPredicate` in the consequent. Therefore, our aim is to design a system which could learn to identify incorrect search steps from minimal user feedback without sacrificing the gains obtained from generalization. This task is complicated by the fact that a typical user may not be able to identify the incorrect search choice(s) made during a multistep reasoning process. The learner should be able to work with delayed feedback about the correctness of the final answer and learn to find plausible inferences for queries. We believe that reinforcement learning is a reasonable method for solving this problem.

Value Iteration Algorithm

```
1. Initialize V(s) arbitrarily
2. Repeat step 3 until policy good enough
3. loop for $s \varepsilon S$
   a. Loop for $a \varepsilon A$
      1. $Q(s, a) \leftarrow R(s, a) + \gamma \sum_{S} T(s,a,s') V(s')$
   b. $V(s) \leftarrow \max_{a} Q(s, a)$
```

Figure 5.4: Value Iteration algorithm (From Kaelbling et al 1996)

Formally, the model consists of (a) a discrete set of states, S; (b) a discrete set of agent actions, A; (c) a reward function R: S x A $\rightarrow$ {-1, 1, 0}and (d) a state transition function T: S x A $\rightarrow \prod(S)$, where a member of $\prod(S)$ is a probability distribution over the set S [Kaelbling *et al* 1996].

In this context, a *state* is the list of predicate types already used during the partially complete search process. At each step of the reasoning process, the inference engine has choice points at which it chooses or rejects different alternatives. It has to assess how useful a particular predicate type is for completing the proof given the predicate types already chosen in the current search path. The *actions* are the selection of a particular predicate type for completing the partial assignment of variables. The *value function* (or V(s)) is the inference engine's current mapping from the set of possible states to its estimates of the long-term reward to be expected after visiting a state and continuing the search with the same policy. Q(*s*, *a*) represents the value of taking the action *a* in state *s*. We use the value iteration algorithm [Kaelbling *et al* 1996] for learning the plausibility of search paths.

For example, V({Group-Topic}) for proving objectFoundInLocation would represent the value of starting with a predicate of type Group-Topic while finding a solution of an objectFoundInLocation-query. Similarly, Q({Group-Topic}, Betweenness-Spatial-Topic) represents the quality of choosing a Betweenness-Spatial-Topic predicate when the partial search path has already chosen a Group-Topic predicate. We use a delayed reward model with user-provided rewards of +1 and -1 for correct and incorrect answers respectively. Rewards are generalized via the predicate hierarchy. For instance, reward statements for spatiallySubsumes are also used for computing V(s) values for its generalizations like spatiallyIntersects. The computational complexity of each iteration of the algorithm is $O(|A||S|^2)$. In our experiments, the policy converged in less than ten iterations. Let us discuss two examples.

Example 1: Q({PhysicalOrderingPredicate}, GeoRegionToGeoRegionPredicate) = 0.99 implies that if we have already selected a PhysicalOrderingPredicate[28] in the first step of the reasoning process, then choosing a predicate of type GeoRegionToGeoRegionPredicate[29] is a good strategy. This is because these predicates combine together to produce correct explanations for objectFoundInLocation-queries.

Example 2: Q({Group-Topic}, LocationPredicate})=0 implies that if we have already selected a Group-Topic predicate in the first step of the reasoning process, then choosing a predicate of type LocationPredicate might not be a good strategy. This is because some predicates of type LocationPredicate lead to incorrect explanations for objectFoundInLocation-queries. For instance, predicates like onPath and pathConnects should not be used while proving objectFoundInLocation-queries [30]. On the other hand, other predicates like

[28] Some examples of PhysicalOrderingPredicate are physicalParts and spatiallySubsumes.
[29] Some examples of GeoRegionToGeoRegionPredicate are easternRegion and geographicalSubRegions
[30] Both onPath and pathConnects are types of LocationPredicate.

objectFoundInLocation can combine with groupMembers[31] to produce correct inference chains for objectFoundInLocation-queries.

5.5 Experimental Method and Results

Exp. No.	Query sets	T	AW	P	R
1	Baseline	833	412	1.00	0.51
	FPEQ	833	412	0.95	0.87
2	Baseline	200	61	1.00	0.42
	FPEQ	200	61	0.92	0.77
3	Baseline	1834	433	1.00	0.32
	FPEQ	1834	433	0.92	0.88
4	Baseline	953	226	1.00	0.34
	FPEQ	953	226	0.93	0.93
5	Baseline	1309	724	1.00	0.43
	FPEQ	1309	724	0.97	0.94

Table 5.1: Summary of Inference Results. Experiment numbers are the same as query numbers

To show that these ideas generate more answers compared to traditional deductive reasoning, we describe a set of experiments. Five sets of questions were selected based on the availability of ground facts in KB and their relevance in learning by reading [Forbus *et al* 2007]. These questions templates were: (1) Where

[31] groupMembers is a type of Group-Topic.

did <*Event*> occur? (2) Who is affected by <*Event*>? (3) Where is <*SpatialThing*>? (4) Who performed the <*Event*>? and (5) Where is <*GeographicalRegion*>? Each question template expands to a disjunction of formal queries. The parameters in the question template (e.g., <*Event*>) indicate the kind of thing for which the question makes sense. Queries were generated by randomly selecting facts for these questions from the KB.

For a baseline comparison, we included all axioms for these predicates and their subgoals through depth 3. We used a simple backchainer working on a LTMS based inference engine [Forbus & de Kleer, 1993]. The depth of backchaining was limited to three and each query was timed out after three minutes. All experiments were done on a 3.2 GHz Pentium Xeon processor with 3GB of RAM. 25% of the queries were used as the training set for learning the V(s) values. Answers whose V(s) values were more than a threshold were accepted.

Table 5.1 compares the performance of FPEQ algorithm and reinforcement learning against the baseline for the test set (i.e. remaining 75% queries). Column **T** is the total number of queries, with **AW** being the number that could be answered given the KB contents, as determined by hand inspection. The columns **P** and **R** indicate precision, and recall, respectively. The user assessed 334 unique answers (from the training set) and the feedback was used for learning the V(s) values. The accuracy of answers provided by FPEQ algorithm was 73%. We then removed answers whose V(s) values were below the threshold. The total number of new answers at this stage is 1010 and the accuracy has improved from 73% to 94%. The FPEQ algorithm mainly reduces false negatives whereas reinforcement learning reduces false positives. Together, they provide a factor of 2.2 improvements (i.e. 120% improvement) over the baseline with an average accuracy of 94%.

It is clear from these results that the ResearchCyc KB contents are not uniformly distributed and different regions have different densities of ground facts. Moreover, some questions are easier to answer than others. For example, the accuracy for Expt. No. 5 is significantly better than the accuracy for Expt. No. 3. We believe that this is due to the fact that it was possible to generate answers for queries involved in Experiment 5 from simple reasoning on a tree-like hierarchy. By contrast, queries involved in Experiment 3 needed more general inference.

As mentioned above, 6% of the derived answers were incorrect. Moreover, the last column in Table 5.1 shows that some answers are still not being generated by the algorithm proposed here. Therefore, we would like to know the types of failures associated with these missed and incorrect answers. Knowledge of the causes of such failures would help the researchers prioritize their research goals. The second column of Table 5.2 (**TC**) shows the total number of corrective actions needed to obtain all answers correctly. It is basically the sum of false positives and false negatives for the algorithm FPEQ. The other columns of Table 5.2 show our by-hand analysis of what additional learning strategies would be required to improve performance further. We have found that randomly chosen training set is imbalanced and leads to redundancy. We believe that a training set which would represent all sections of the KB without redundancy would be smaller and lead to better results. The third column (labeled **A**) in Table 2 shows the number of problems which would be solved by a better training set. In some cases, we found that we need to augment the graph representation so that it could handle functions (e.g. `(industryFacilities (IndustryOfRegionFn OilIndustry SaudiArabia) Iraq-SaudiArabiaPipeline)`), and quantified assertions. The number of such cases is shown in the column labeled **B**. The column labeled **C** shows cases when additional knowledge would have helped. Cases where a PIP needs to be replaced by a more specific pattern in terms of existing predicate types are indicated by **D**, and cases where a new predicate type between existing predicate types would improve matters are indicated by **E**. Note that the amount of training data is small compared to the relative improvement for each experiment.

Exp. No.	**TC**	**A**	**B**	**C**	**D**	**E**
1	73	47	0	5	19	2
2	18	14	0	0	4	0
3	81	12	1	39	28	1
4	33	14	2	0	16	1
5	66	9	1	33	22	1

Table 5.2: Distribution of Learning Actions.

Conclusion

Learning to reason, while minimizing user intervention, is an important problem. We have shown how plausible inference patterns, expressed in terms of higher-order knowledge and learned via reinforcement learning, can be used to reason with reasonable accuracy. The use of predicate types for representing PIPs leads to a succinct, easily learnable and tractable representation. The FPEQ algorithm mainly reduces false negatives whereas reinforcement learning reduces false positives. By integrating them, we get a 120% improvement over the baseline with an average accuracy of 94%.

While these experiments used the contents of ResearchCyc, we believe they would be applicable to any large-scale KB whose predicate types were classified sensibly. Our technique is especially suitable for knowledge capture because it exploits ground facts, which are much easier to gather than logically quantified facts. We believe that this technique can be used to help bootstrap intelligent systems and reduce the dependence on hand-crafted axioms.

Our results suggest three lines of future work. First, we found that a randomly chosen training set does not represent all regions of the KB adequately. Finding these gaps in coverage could be used to suggest new learning goals for learning by reading and other forms of knowledge capture. Second, being able to refine plausible inference patterns to use more specific predicate types would improve accuracy and coverage. Finally, plausible inference patterns could be used as an intermediate stage for postulating new logically quantified statements, perhaps by using a technique like relational reinforcement learning approach [Dzeroski *et al* 2001] to carry out the refinements.

6. Modeling the Evolution of Knowledge in Learning Systems

6.1 Introduction and Motivation

In recent years, there has been considerable interest in Learning by Reading [Barker et al 2007; Forbus et al 2007, Mulkar et al 2007] and Machine Reading [Etzioni et al 2005; Carlson et al 2010] systems. The study of these systems has mainly proceeded along the lines of measuring their efficacy in improving the amount of knowledge in the system. Learning by Reading (LbR) systems have also explored reasoning with learned knowledge, whereas Machine Reading systems typically have not, so we focus on LbR systems here. These are evolving systems: Over time, they learn new ground facts and new predicates and collections are introduced, thereby altering the structure of their knowledge base (KB). Given the nascent state of the art, so far the learned knowledge is typically small compared to the knowledge base the system starts with. Hence the size of the KB is constant for all practical purposes, and the set of axioms it uses for reasoning will be stable and continue to perform as they did before. But what will happen to reasoning performance as the state of the art improves, and the number of facts the system has learned by reading (or using machine reading techniques) dwarfs its initial endowment?

To explore such questions, we introduce an *inverse ablation model*. The basic idea is to take the contents of a large knowledge base (here, ResearchCyc[32]) and make a simulation of the initial endowment of an LbR system by removing most of the facts. Reasoning performance is tested on this initial endowment, including the generation of learning goals. The operation of a learning component is simulated by gathering facts from the ablated portion of the KB that satisfy the learning goals, and adding those to the test KB. Performance is then tested again, new learning goals are generated, and the process continues until the system converges (which it must, because it is bounded above by the size of the original KB). This model allows us to explore a number of interesting questions, including: (1) How does the growth in the number of facts affect reasoning performance? (2) How might the speed at which different kinds of concepts are learned vary, and what factors does that depend upon? (3) Is learning focused, or are we learning facts about a wide range of predicates and concepts? (4) What are the

.

[32] http://research.cyc.com

properties of different learning strategies? (5) How does the distribution of facts that can be acquired affect the learning trajectory?

The inverse ablation model provides a general way to explore the evolution of knowledge bases in learning systems. This chapter describes a set of experiments that are motivated by LbR systems. Under the assumptions described below, we find that (1) the size of the KB rapidly converges, (2) the growth is limited to a small set of concepts and predicates, spreading to only about 33% of the entire growth possible, (3) different concepts show different rates of growth, with the density of facts being an important determining factor, and (4) Different learning strategies have significant differences in their performance, and the distribution of facts that can be learned also plays an important role.

The rest of this chapter is organized as follows: We start by summarizing related work. A detailed description of the inverse ablation model and experimental results are described next. In the final section, we summarize our main conclusions.

6.2 Related Work

A number of researchers have worked on Learning by Reading and Machine Reading systems. Learning Reader [Forbus et al 2007] used a Q/A system for evaluating what the system learned, and included *rumination.* Mobius [Barker et al 2007] was evaluated by comparing the facts produced by their system to a manually-generated *gold standard* set of facts. NELL [Carlson et al 2010] also uses human inspection to evaluate the quality of the knowledge produced. These systems all produce formal representations. In contrast, TextRunner [Etzioni et al 2005] produces word-cluster triples. A prototype system for deriving semantic representations of sentences for two domains has been discussed in [Mulkar et al 2007]. Experiments related to populating the Cyc KB from the web have been described in [Matuszek et al 2005]. These systems have provided useful insights for improving our understanding of learning systems. However, measurements involving the temporal evolution of KBs and the systemic properties of rapidly changing learning systems have not been the focus of these endeavors. In addition to LbR research, our work is inspired by the literature on the evolution of the World Wide Web [Ntoulas et al 2004], graphs [Leskovec et al 2007] and social networks [Kossinets & Watts 2006].

6.3 An Inverse Ablation Model

Deductive reasoning is a primary reason for accumulating large knowledge bases. In large knowledge-based systems, inference engines generate and examine thousands of potential proof paths for answering target queries. Understanding how deductive inference performance changes as KBs grow is the fundamental motivation for the inverse ablation model. Since large-scale learning systems are in their

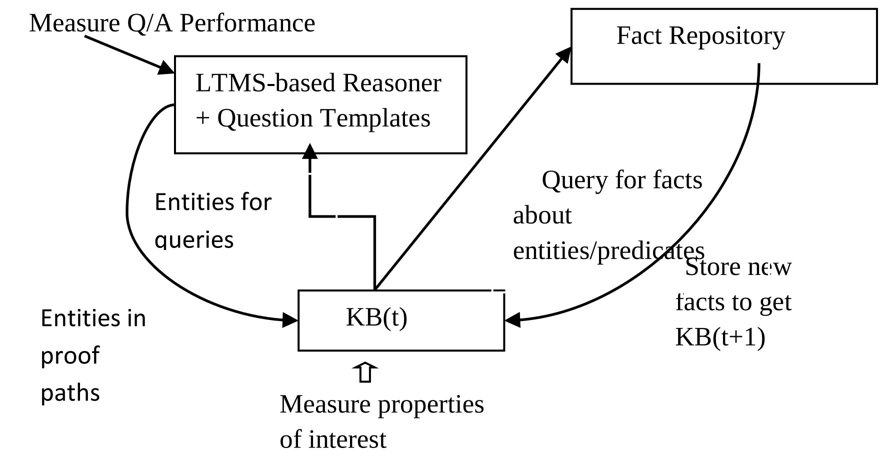
Measure Q/A Performance
Fact Repository
LTMS-based Reasoner + Question Templates
Query for facts about entities/predicates
Entities for queries
Store new facts to get KB(t+1)
Entities in proof paths
KB(t)
Measure properties of interest

Figure 6.1: Inverse Ablation model

infancy, instrumenting a learning system that is operating over months is still not possible. Hence we start by ablating a large KB and measure reasoning performance as we add knowledge back in. Figure 6.1 shows a schematic diagram of how the inverse ablation model works. The parameters of an inverse ablation model include (1) What is the initial endowment? (2) What reasoning methods are used?, (3) How are queries generated?, (4) What is the distribution of facts in the external knowledge source?, and (5) What is the strategy used to grow the knowledge base? We discuss each decision in turn.

***Initial endowment*:** Since we are using ResearchCyc contents, the initial endowment consists of the basic ontology definitions (the `BaseKB` and `UniversalVocabularyMt` microtheories) plus about 5,000 facts chosen at random. This leaves 491,091 facts that could be added on subsequent iterations to simulate learning. We refer to this collection of facts as the *fact repository*, to distinguish it from the KB used in reasoning during a learning iteration. One interesting measure is how much of the fact repository ends up being added back when the system converges: Facts that remain in the repository at that point have no perceived relevance to the questions that are driving learning.

***Reasoning method*:** CSP solvers are arguably the most efficient solvers available today, but are largely limited to propositional reasoning, making them inappropriate for open domains and large-scale worlds where propositionalization would lead to an exponential explosion in the number of axioms. By contrast, Cyc systems include broadly capable reasoners that handle a wide variety of higher-order constructs and modals, making them very flexible, at the cost of efficiency. The reasoning system we use here is FIRE because it was used in the Learning Reader system [Forbus et al 2007]. FIRE performs backchaining over Horn clauses, similar to Prolog but without clause ordering or cut, and uses an LTMS [Forbus & de Kleer 93] for caching answers. Following Learning Reader, inference is limited to depth 5 for all queries, with a timeout of 90 seconds per query. Each parameterized question template is expanded into a set of formal queries, all of which are attempted in order to answer the original question.

***Query Generation*:** We automatically generate a set of queries at each iteration by asking every question for every entity that satisfies the collections associated with each type of parameterized question. Thus the types of entities, given the set of parameterized questions, are Event, Person, and GeographicalRegion. Note that as the KB grows, so too can the number of queries generated, since new entities of these types can be added. This allows us to measure how costly different strategies for generating learning goals might be.

***Growth Strategy*:** The method for growing the KB by adding back in facts should reflect assumptions made about the way the system generates learning goals. Moreover, it is also interesting to study the properties of different learning strategies. Below we compare the performance of two strategies:

(1) *Entity-based Learning Strategy*: At each iteration, we use reasoning failures to generate learning goals, which are then used to gather facts from the fact repository. Specifically, the proof trees for failed queries are examined to find nodes representing queries involving specific entities. Finding out more about these entities become the learning goals for that iteration. For example, a query like `(acquaintedWith BillClinton ?x)` leads to an intermediate query like `(mother ChelseaClinton ?x)`. Hence learning about `ChelseaClinton` would become one of the learning goals for that iteration. We model the effect of learning by gathering all of the facts which mention the entities in learning goals from the fact repository. This is tantamount to assuming a large amount of learning effort in every cycle, essentially mining out everything that is going to become known

about an entity the first time that it becomes a target for learning. While optimistic, pursuing any other strategy would require making more assumptions, thereby making them harder to justify. This gives us an extreme point, at least.

(2) *Predicate-based Learning Strategy*: While using this strategy, the reasoner chooses a new predicate *pred* in every learning iteration, which would lead to maximum improvement in Q/A performance. All facts matching the pattern `(pred ?x ?y)` are sought from the fact repository. The algorithm used for assessing the utility of learning about predicate *m* is shown in Figure 6.2. Here NumberOfFacts(p) represents our estimate of the number of facts which can be inferred about predicate p. In step 1 of the algorithm, we initialize it by KBFacts(p), which represents the number of ground facts

```
Algorithm: ReturnEstimateOfPerformance

Input: Predicate m

1. For all predicates p in the KB do:
      a. NumberOfFacts(p)←KBFacts(p)

2. NumberOfFacts(m)←KBFacts(m)+N

3. OrderedList ← Perform a topological sort of the search space represented by the
   axioms. Break cycles arbitrarily.

4. For each p in OrderedList
     $NumberOfFacts(p) \leftarrow \sum_Q NumberOfFacts(q) + \sum_R \prod_S \alpha * NumberOfFacts(s)$

5. Return $\sum_{RootNodes} NumberOfFacts(r)$
```

Figure 6.2: Algorithm for assessing the utility of learning (see text for explanation)

in the KB. about predicate *p*. In step 2, we assume that we can get N facts (In this work, N was set to 1,000.) from the fact repository about predicate *m*. A topological sort of the search space is performed in step 3 and inference is propagated bottom-up in step 4. The first-term on the RHS[33] of step 4(a) gathers evidence from the OR nodes which are the children of p. The second term gathers evidence from the AND children of p. The constant α represents the probability of unification and was set to 10^{-3}. Step 5 returns the number of answers for the root nodes (i.e., the target queries). The process is repeated for all predicates and the predicate which return the maximum value is sent to the fact repository as the learning query.

Figure 6.3 describes the experimental procedure used, in algorithmic form. Step 6 describes the entity-based learning strategy, while Step 7 describes the predicate-based strategy.

```
Algorithm

Input: Growth Strategy (Entity-based or Predicate-based)

4.  Set t← 0.
5.  Initialize KB(t) by choosing facts randomly from the repository.
6.  Repeat steps 4 to 9 until the process converges
7.  Set Q ← Generate all questions for the question templates mentioned on page 2.
8.  Ask the set of questions Q and measure Q/A performance.
9.  If the Growth Strategy is Entity-based then:
    a. E ← the set of entities in intermediate queries generated during the reasoning process.
    b. Let Facts ← New facts about the elements of E in the Fact Repository.
10. Else if growth strategy is Predicate-based
    a. Choose a predicate p from the search space which would lead to the maximum gain in
       Q/A performance.
    b. Let Facts ← New facts which match the pattern (p ?x ?y) from the Fact
       Repository.
11. KB(t+1) ← KB(t) + Facts
12. Record the properties of interest for KB(t+1)
13. If ΔKB → 0 then exit loop, else t ← t+1and go to step 4.
```

Figure 6.3: Inverse Ablation Model

[33] The sets Q and R are the OR and AND children of p respectively. For a given AND node, the set S represents its antecedents.

Distribution of Facts in the External Knowledge Source: The trajectory of learning depends on the distribution of facts in the external knowledge source. Note that since we are sending entities and predicates to the fact repository, we have shown the probability of number of facts per entity and number of facts per predicate in Figure 6.4. The last bin in Figure 4 shows the probability mass not included in other bins (i.e. Pr(x> 50)).

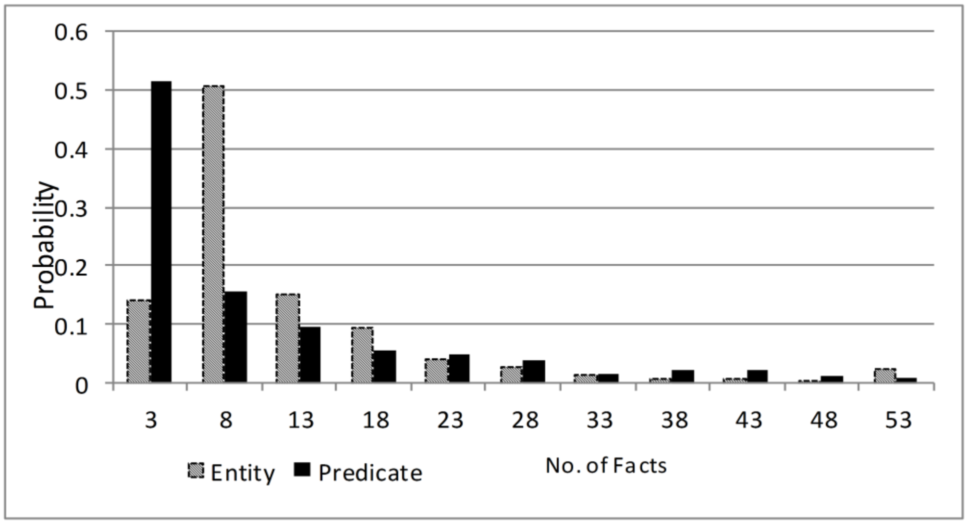

Figure 6.4: Distribution of facts in external knowledge source

6.4 Experimental Results

We experimented with three starting points for KB(0). Since the results for these experiments were similar, we report average of these results in Figures 6.5 to 6.10. Figure 6.5 shows the change in number of ground facts. For the entity-based model, the number of facts increases rapidly from 4,968 at t=0 to 143,922 facts at t=2. The curve asymptotes to about 166,000 facts at t=5. It is also useful to compare the extent of this growth with respect to the contents of fact repository. The coverage increases from 1% of the repository at t=0 to 33% at t=5. The high rate of growth shows that the domain is densely connected and the average distance between two nodes is pretty small. On the other hand, given these questions, about 67% of the repository is beyond our reach. The number of facts asymptotes at 5% of the fact repository for the predicate-based model.

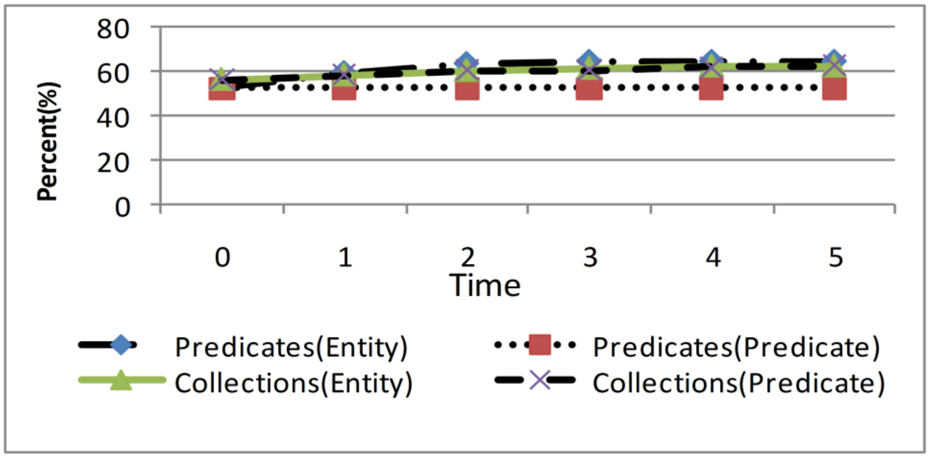

Figure 6.6: Change in the number of new predicates/concepts (Average of three experiments.)

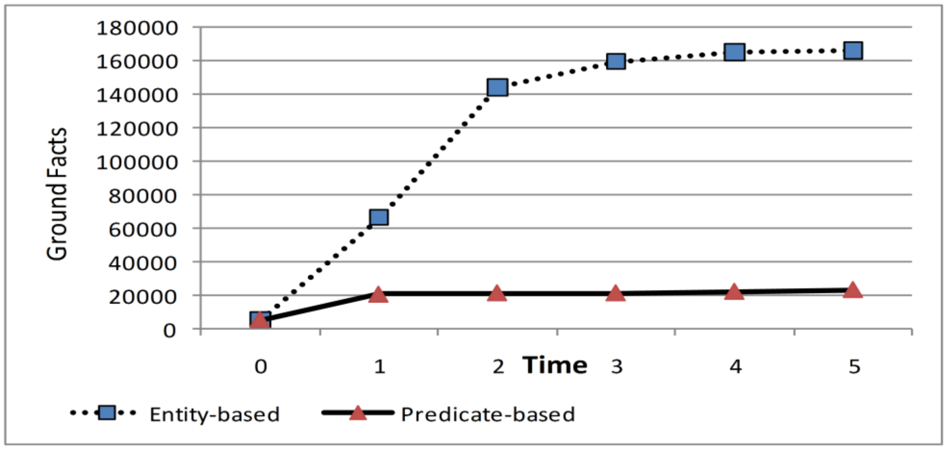

Figure 6.5: Change in number of ground facts (Average of three experiments.)

Next, we turn to the rate of introduction of new predicates and concepts (see Figure 6). In this case, both learning strategies showed similar performance. At t=0, about 55% of the predicates had least one ground fact associated with them. After five learning iterations, 65% predicates had at least one ground fact[34]. Similarly, the proportion of concepts with at least one instance increased from 53% to 62%. This shows that the learning is focused and new facts are being drawn from a small set of predicates and concepts. It also points towards homophily in the ground facts because many different concepts are out of our reach.

In Figure 6.7, the dynamics of Q/A performance is shown. The proportion of questions answered improves significantly with the size of the KB. For the entity-based model, the size of KB increased by 3,104% in five iterations, but the proportion of questions answered increased by only 637%. The time needed per query increased by 533% during this period. These results suggest that time-constrained deductive reasoning systems will need new methods to select the best set of axioms due to increasing resource requirements and changing distribution of facts and collections. The entity-based model performs better than the predicate-based model as far as net Q/A coverage is concerned. On the other hand, Figure 6.8 shows that the predicate-based model uses fewer facts to derive its answers than the entity-based model. The number of question answered per ground fact also improves with time for both strategies.

[34] This refers to the number of predicates at the top-level.

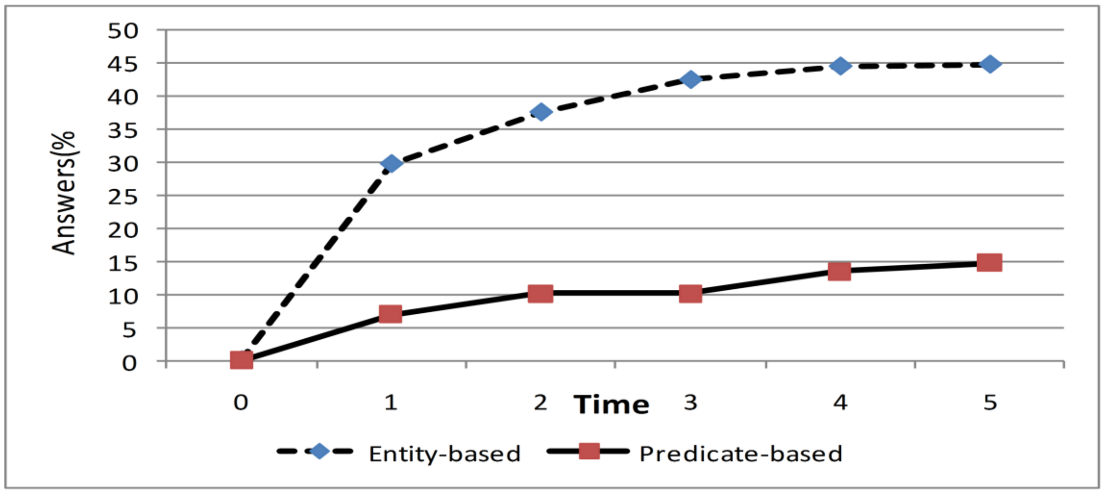

Figure 6.7: Q/A performance (Average of three experiments)

It is also interesting to compare the rate of growth of different regions of the KB and check if some of them display unusual patterns. Recall that the question types discussed involve three kinds of concepts: `Person`, `Event` and `GeographicalRegion`. The predicate-based model did not show any great difference in growth patterns in different regions of the KB (see Figure 6.9). However, the rates of growth of instances of these concepts vary greatly for the entity-based model. In the figure below, we see that the KB had 1.4% of all instances of `Person` at t=0. This grew to 2% after five iterations. During the same period, the proportion of `GeographicalRegion` increased from 7.9% to 58%. The proportion of instances of `Event` grew from 26% to 33% (not shown in Figure). It shows that the rate of growth of `GeographicalRegion` is high, whereas this model has not made significant progress in accumulating

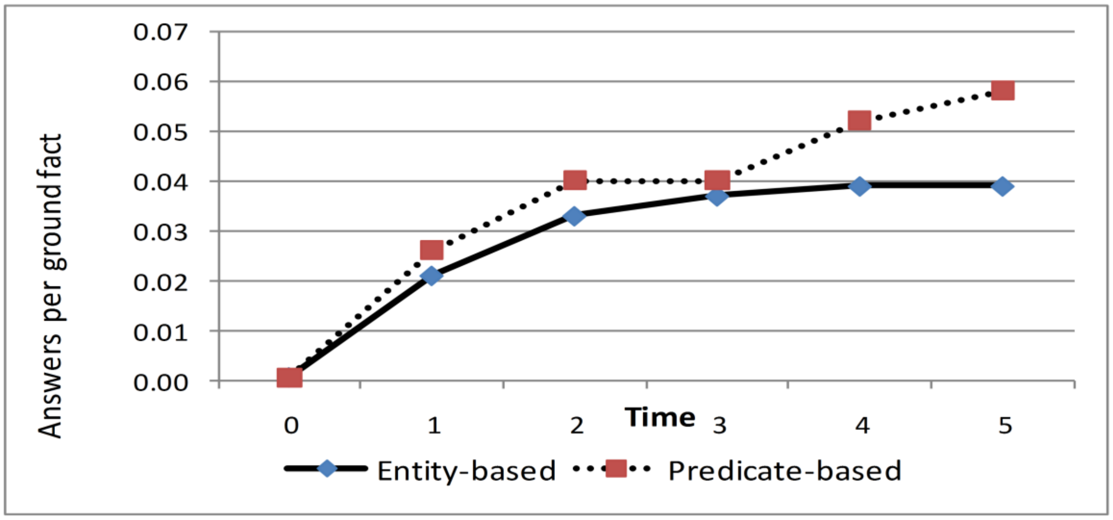

Figure 6.8: Number of answers per unit ground fact (Average of three experiments)

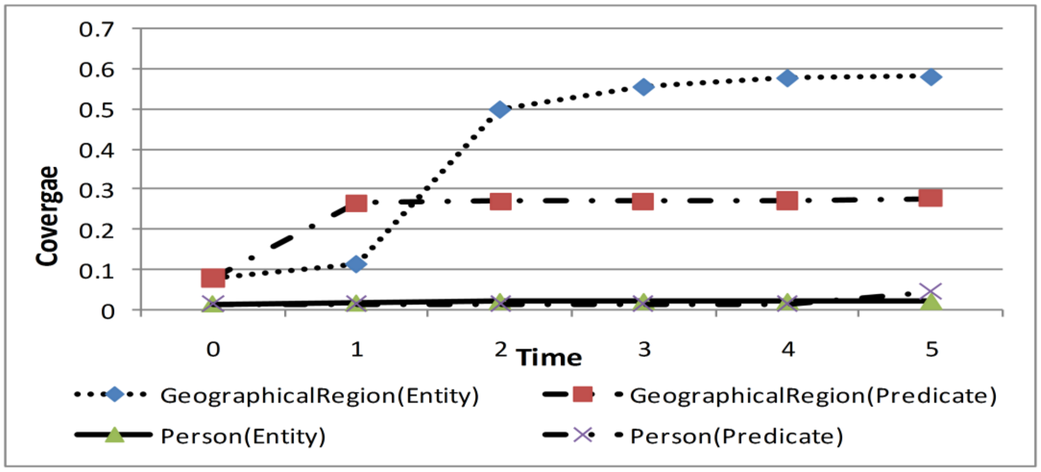

Figure 6.9: Growth of different concepts (Average of three experiments)

knowledge about instances of `Person`. One important reason for this difference is the density of facts for these concepts. In Figure 6.10, we show the distribution of number of facts per entity for these concepts. The x-axis shows the number of facts per entity for instances of each of these three concepts. The mean of facts per entity for `Person`, `Event` and `GeographicalRegion` are 2.14, 5.58 and 11.29 respectively. The medians of facts for these concepts are 1, 2 and 5 respectively. The net growth in coverage for these concepts was 0.5%, 6.2% and 50.1% respectively. This shows that the density and the rate of growth show a nonlinear relationship and it can be used to modulate the rate of learning.

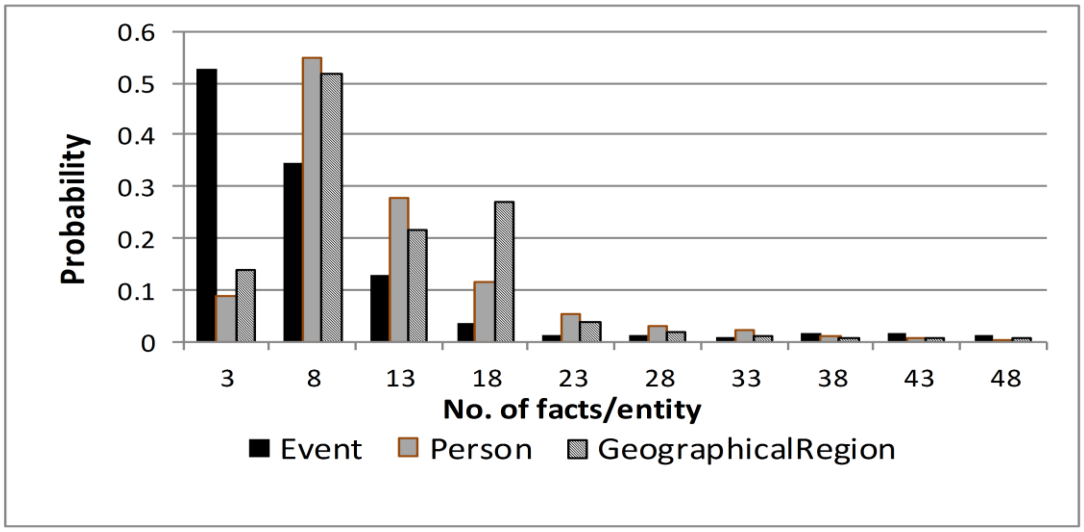

Figure 6.10: Probability distribution of facts/entity for concepts

In Table 8.1, we summarize key differences between entity-based and predicate-based strategies.

	Entity-based	**Predicate-based**
No. of Facts	33% of maximum	5% of maximum
Focused Learning	Yes	Yes
Q/A (%)	Better	Worse
Utilization of ground facts for deriving answers	Worse	Better
Distribution of Learning	Skewed	Uniform

Table 8.1: Summary of key differences

6.5 Conclusion

There has been growing interest in creating large-scale learning systems, such as Learning by Reading systems. However, there has been relatively little work in studying the properties of reasoning systems which grow significantly over time. We have proposed an inverse ablation model for studying how reasoning performance changes with KB growth, as might be caused by learning. The method proposed here is very general and could be used with any large KB or KR&R system.

We have studied the performance of two learning strategies that are of particular interest from the perspective of learning systems. There are significant differences in their properties. These differences are due to the different probability distributions of entities and predicates in the external knowledge source (Figure 8.4). Understanding how best to combine these strategies and use the distribution of facts to achieve a balanced and efficient learning trajectory is an interesting open question. We observed that one of the models proposed here increased the size of the KB from 1% to 33% of the repository in five iterations. As the number of facts, predicates and collections increase, the size of search space and dynamics of reasoning would change as well. This implies that learning algorithms and inference engines should use distribution-sensitive algorithms in order to adapt well to a changing KB. Growth is compartmentalized but spreads to a significant fraction of the fact repository. Growth is focused, as indicated by the new facts being about a small number of predicates and concepts. Different concepts show different rates of growth, which can be explained by their densities. Our results show that the rate of growth in high density regions is very high. Since the optimal rate of growth varies from application to application, some systems may need to find appropriate parameters for controlling growth in high density regions. On the other hand, increasing the knowledge about low density regions is a challenge. In a sparsely connected domain, learning systems may need to find ways to hop from one island to another using other learning methods. Applying these learning strategies in a new learning by reading system is something we plan to do in future work.

7. Growth Patterns of Inference

7.1 Introduction and Motivation

In recent years, there has been considerable interest in Learning by Reading [Barker et al 2007; Forbus et al 2007, Mulkar et al 2007] and Machine Reading [Etzioni et al 2005; Carlson et al 2010] systems. Such systems are already good at accumulating large bodies of ground facts (although learning general quantified knowledge is currently still beyond the state of the art). But will the new ground facts learnt by them help in improving deductive reasoning? Ideally, new facts should lead to improvements in deductive Q/A coverage, i.e. more questions are answered. Will the rate of performance improvement always be uniform, or will there be “phase changes”? Understanding the dynamics of inference is important to answering these questions, which in turn are important for making self-guiding learning systems.

Our analysis draws upon ideas from network analysis, where the networks are the AND/OR connection graph of a set of first-order Horn axioms. By analogy to epidemiological models, we explore diffusion of inference in the network, i.e. how does coverage of queries increase as new ground facts are learned. Cascade conditions correspond to when inference becomes easy, i.e. increased coverage. Here we argue that some useful insights about growth patterns of inference can be derived from simple features of search spaces. We focus on three parameters: The first, α, associated with each node, represents the contribution of each node in answering a set of questions. Parameters *k* and *β*, defined below, represent the connectivity of the graph. We study Q/A performance for different values of these parameters, including several sizes of KB contents, to simulate the impact of learning. We found that search spaces with skewed degree distribution lead to better Q/A performance in smaller KBs, whereas in larger KBs more uniform search spaces perform better. In some cases, as α increases, the percolation of inference shows a significant and abrupt change. A degenerate case, in which the effect of ground facts “dies down” and expected improvements in Q/A performance are not observed due to mismatch of expectations and ground facts, is also seen.

The rest of this chapter is organized as follows: We start by summarizing related work. A detailed description of the diffusion model and experimental results are described next. In the final section, we summarize our main conclusions.

7.2 Related Work

In social sciences, there has been significant interest in models of different kinds of cascades. In these domains, the interest is to study how small initial shocks can cascade to affect or disrupt large systems that have proven stable with respect to similar disturbances in the past [Watts 2002]. The model described here is inspired by work on cascades in random graphs [Watts 2002] and epidemic thresholds in networks [Chakrabarti et al 2008]. In AI, there has been work on viral marketing [Domingos & Richardson 2001] and phase transitions in relational learning [Giordana & Saitta 2000], who uses somewhat similar parameter definitions to ours. However, neither of them addresses deductive reasoning in first-order knowledge bases, as we do.

7.3 A Model for Spread of Inference

How does the possibility of inference cascades depend on the size of KB and the network of interconnections? In an inference cascade, the effects of new ground facts reach the target queries quickly, and a small number of new facts should lead to disproportionate effects on the final Q/A performance.

To study these issues, it is useful to view a learning system as a *dynamical system*, where the *state* of the system is given by a set of parameters. In what follows, we define some parameters which are useful for describing inference in a knowledge-based learning system. Then we report experimental results for different values of these parameters. It follows that the aim of a meta-reasoning module should be to identify more desirable states of such a system and use this information for its guidance.

Now, we describe our model of inference propagation. As discussed in Chapter 2, when answering a parameterized question, each template expands into a set of formal queries, all of which are attempted in order to answer the original question. The FIRE reasoning system uses backchaining over Horn clauses with an LTMS [Forbus & de Kleer 93]. Starting with the queries for the 10 parameterized question types, we find all Horn clauses relevant for answering them and continue this process recursively for their children until depth 10. This leads to a set of 7,330 first-order axioms. We use different subsets of this axiom set for studying the properties of search spaces.

The graph G is acyclic AND/OR graph generated during the backward chaining of rules. Let N be the set of nodes in this graph. We explore variations in the structure of the search space by choosing subsets of N, say M, and keeping only the edges of G which directly connect nodes in M. In order to consider the space of inferences that could be done with a search space, we define Q to be the set of specific parameterized questions which could be asked for all 10 of the questions defined in Chapter 2. That is, the variable representing the parameter (e.g. <*Event*>) is bound to all possible entities in the KB of that type, whereas the other parameter in each query remains open, to be solved for. For every node m, *depth(m)* represents its depth in G. We can now define α as follows:

$$\alpha = \frac{1}{|N|} \sum_{m \in M} \frac{Solutions(m)}{|Q| * (depth(m) + 1)}$$

Solutions(m) represents the number of answers returned by the node *m* on its own (i.e., purely by ground fact retrieval and not by using axioms). α represents the average contribution of each node towards answering the set of queries. Depth of nodes has been used to weigh the contribution of nodes because solutions closer to the root node are more likely to percolate up due to fewer unification problems. Another factor which plays an important role in determining the diffusion of inference in search spaces is their connectivity. The larger the degree of nodes, the more likely it is to return a solution by using answers from its neighbors. Moreover, since different degree distributions lead to significant differences in the properties of dynamics of networks, we study how they affect the rate of percolation of inference in the search space.

Model 1: We start with the set of 7,330 axioms discussed above. We begin with the target queries and choose *k* children for them at random. If the node has less than *k* children, we choose all of its children. This is done recursively for the children nodes until depth 10. In other words, by construction, the degree of all nodes is less than or equal to *k*. For example, let us consider the search space shown in Figure 8.2. Then Figure 8.3 and 8.4 show examples of search spaces which could be generated from this model when *k* is 2. (Colored nodes in these figures represent selected nodes.)

Model 2: We start with a set of 7,330 axioms discussed above. We begin with target queries and choose β% children for them at random. This is done recursively for the children nodes until depth 10. For example, if β=50, then Figure 8.5 shows an example of search space generated from this model.

In this work, we study the properties of two types of degree distributions: (i) Uniform distribution and (ii) Skewed distribution (i.e., resembling scale-free). To generate search spaces from these distributions, we do the following. We start with the axiom set defined in the previous section. By selecting different subsets of these axioms, we can understand how the structure of search space affects inference. As we select different subsets, we are accessing different regions of KB. Since facts are not uniformly distributed in the KB, α for these axiom subsets varies significantly. Moreover, axioms in ResearchCyc KB have a skewed degree distribution. In other words, most predicates have very few children in the query graph, whereas there are many axioms for proving a small number of predicates. See Figure 7.2 for a simplified example. (For simplicity, only OR nodes have been shown.) A description of two methods for generating these search spaces is shown in Figure 1. Model 1 makes the search space more uniform by limiting the maximum number of children to *k*. On the other hand, Model 2 preserves the skewed degree distribution by selecting β% children of each node. Which of these models is better for better Q/A performance? Moreover, how does the performance depend on *k* and β? We believe that these parameters (i.e., α, β and *k*), play an important role in understanding the dynamics of inference and the possibility of inference cascades in deductive search spaces. To understand the performance differences between these search spaces, we generated a number of instances of these models. For Model 1, we generated axiom sets for $2 \leq k \leq 7$. Similarly, for Model 2, we generated axiom sets where β ε {10, 15, 20, 30, 40, 50}. For each value of *k* and β, at least seven sets of search spaces were generated[35].

[35] In some cases, we generated more axiom sets to locate the transition point more precisely. For example, see Fig 7.11 and 7.13.

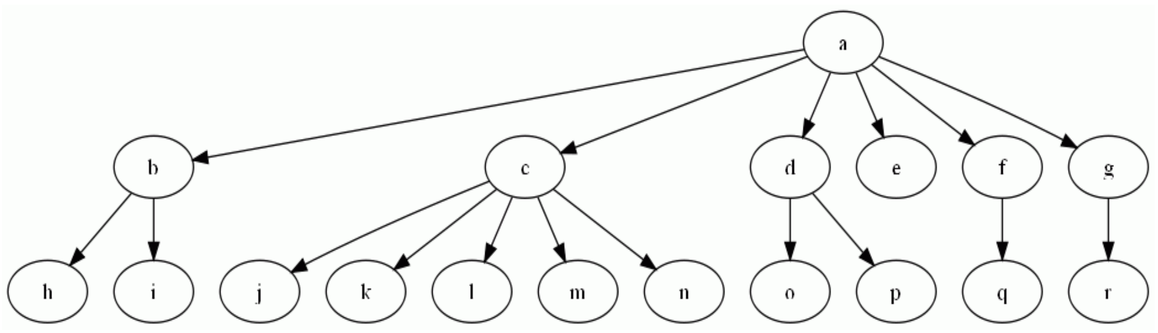

Figure 7.2: An example of a search space with skewed degree distribution.

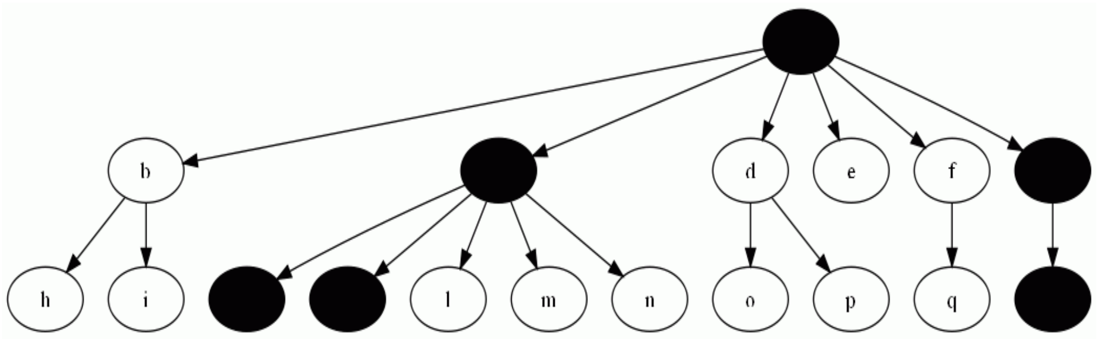

Figure 7.3: An example of Model 1 search space. Here *k*=2.

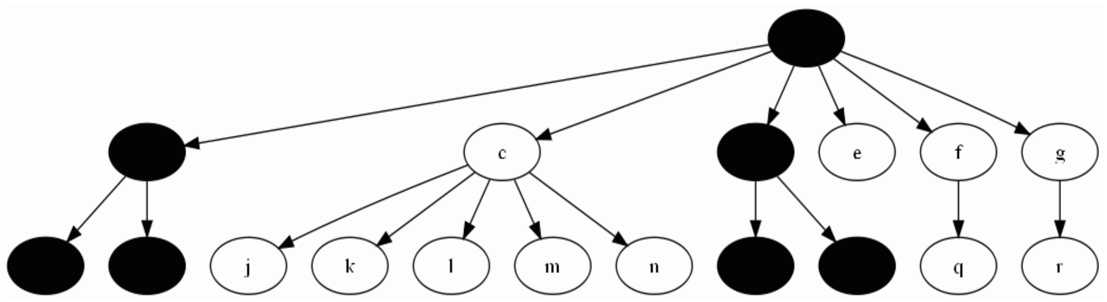

Figure 7.4: Another example of Model 1 search space. Here *k*=2.

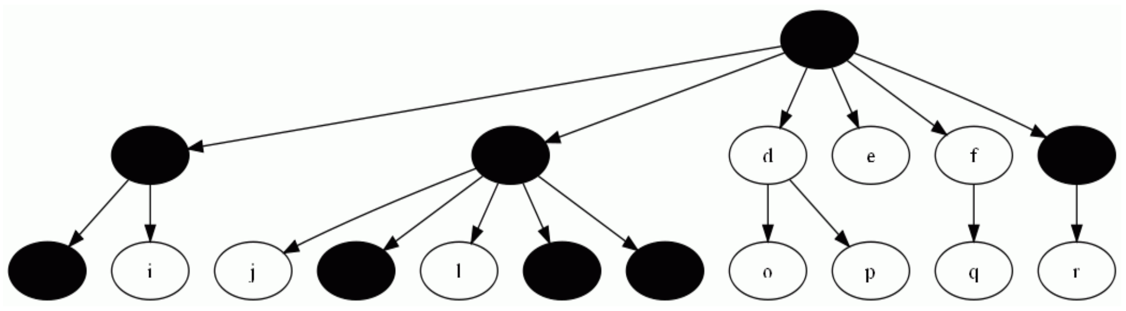

Figure 7.5: An example of Model 2 search space obtained from the search space shown in Figure 7.2. (β=50).

As learning systems gather thousands of facts from reading and other sources, the size of their KBs will grow. To model and understand the effects of increasing KB size on the performance of learning systems, we use the inverse ablation model [Sharma & Forbus 2010]. The basic idea is to take the contents of a large KB (here, ResearchCyc) and create a small KB by ablation and incrementally re-add facts, collecting snapshots of reasoning performance of the system. We use this method to generate two KBs of size 705,180 and 865,992 facts respectively. We also use the original ResearchCyc KB which contains 1.2 million facts Therefore, the model presented here has been evaluated on these three KBs. In what follows, they are referred to as KB_1, KB_2 and KB_3 respectively.

It is obvious that as α increases we should be able to answer more questions. However, we found that in about 28% of all cases, search spaces with high α did not lead to expected high performance. This abnormality is mainly seen when minimal unification takes place at a small number of nodes which lie in all paths from the leaves to the root nodes. An example of this phenomenon is shown in Figure 7.6. We see how inference propagates from the

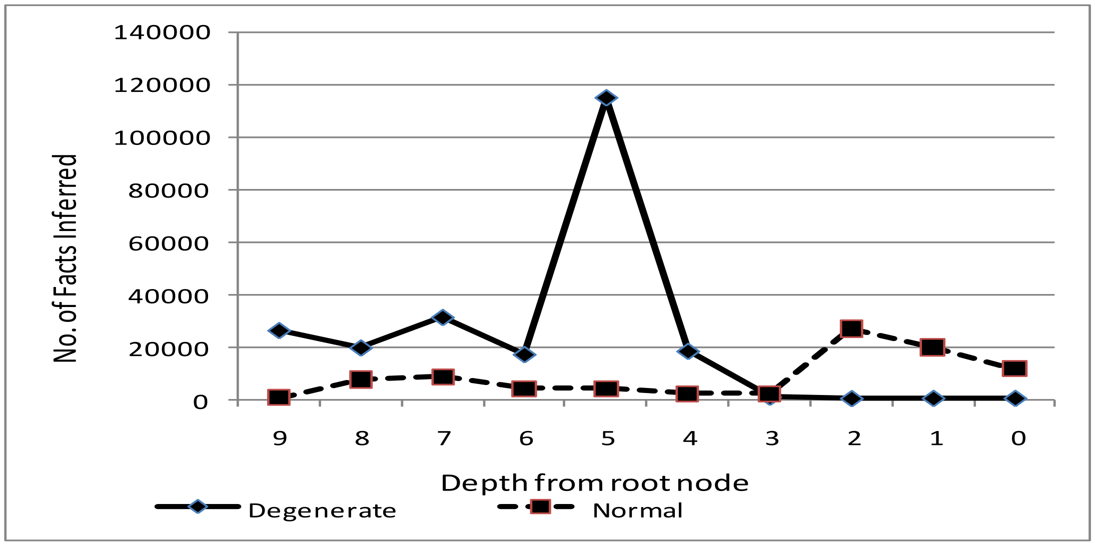

Figure 7.6: An example of a degenerate case

leaves (depth 9) to the root nodes (depth 0). We observe that for the degenerate case, about 120,000 facts are inferred at depth 5. However, this was reduced to almost zero answers at depth 3. The other search space accesses less ground fact rich regions (about 27,000 facts at depth 2), but manages to produce more than 10,000 answers at the root node. This shows that small mismatch between the expectations of axioms and ground facts can lead to serious problems in inference propagation. In this particular degenerate case, no unification took place at the node which joined the ground fact rich regions to the root node. Although the models of selecting axioms described here are admittedly simple, it is surprising that 28% of all axiom-sets generated by them have such serious problems. This implies that knowledge acquisition process has to be informed of the expectations of inference chains. In the absence of such a process, the effects of thousands of facts learnt from different sources would degenerate, and the results of learning would not show up in the final Q/A performance.

7.4 Experimental Results

In this section, we study how α, β and k affect the dynamics of Q/A performance. Recall the set of 10 questions discussed before. All questions which satisfy the constraints of these templates were generated.

KB_1, KB_2 and KB_3 led to 5409, 13938 and 36,564 queries respectively. We investigated the following questions: (1) How does Q/A performance change as we access those regions of KB which have more facts (i.e., α increases)? (2) What is the nature of Q/A performance as search spaces become denser (i.e., as k and β increase) and (3) Under what conditions does inference percolate to a sizeable section of the KB helping us to answer more than a given threshold fraction of questions? (In this work, the threshold is 0.2)

Figure 7.7 shows the average performance of Model 1 search spaces for the three KBs. We observe that the threshold performance was not reached for any value of k in the smallest KB. On the other hand, as k increases, performance gradually improves. Moreover, as KB becomes bigger, the threshold is achieved for sparser search spaces. In Figure 7.8, we see similar trends for Model 2 search spaces. The only significant difference is that search spaces for $\beta > 30$ attain threshold performance in KB_1 as well. Figure 7.7 and 7.8 show that larger KBs lead to better Q/A performance even with fewer axioms. It is interesting to note that there is a very small difference in the performance in different KBs for higher values of β in Model 2 search spaces, whereas their performance varies significantly in Model 1 search spaces. This brings us to another interesting question: Which of the two models discussed here lead to better Q/A performance? We note that although higher values of k and β imply higher connectivity, there is no one-to-one correspondence between these parameters. Therefore, to compare these two models, we selected a set of axiom-sets which had same average degree from both models. The average number of answers for these axiom-sets was then measured. The results are shown in Table 7.1. We see that while uniform search spaces perform better for larger KBs (i.e., KB_2 and KB_3), search spaces with skewed degree distribution perform better in smaller KBs (i.e., KB_1).

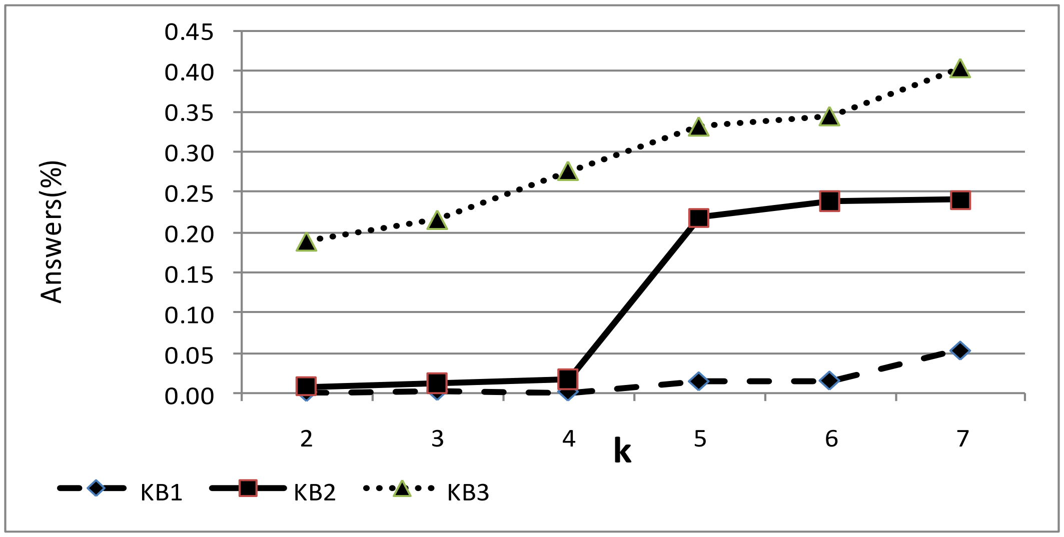

Figure 7.7: Q/A performance for Model 1 search spaces.

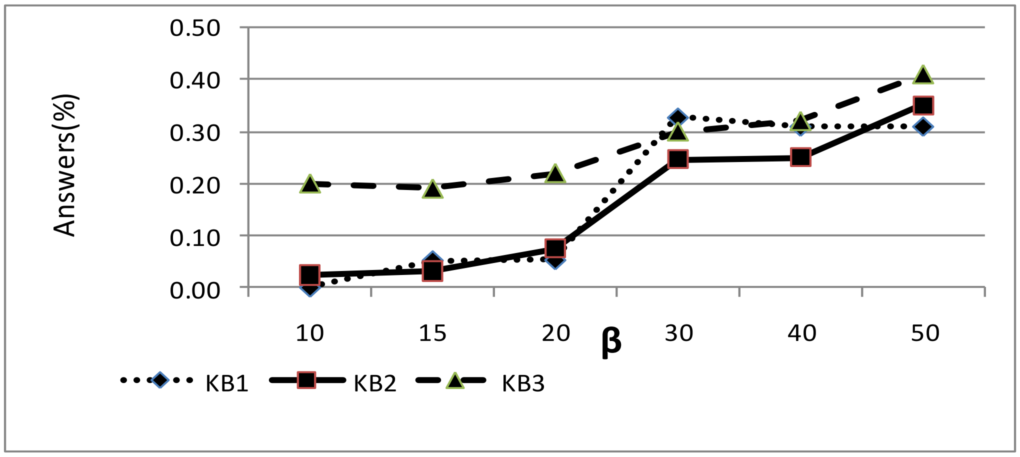

Figure 7.8: Q/A performance for Model 2 search spaces.

KB	Model 1	Model 2	Change in Model 2 w.r.t. Model 1
KB_1	33	50	+51.5%
KB_2	840	213	-74.6%
KB_3	10176	9512	-6.5%

Table 7.1: Average number of answers for two models

Next, we discuss the change in Q/A performance as α increases. How does the performance change as we access those regions of KB which are richer in facts? It is clear that as α and the density of search space increase, the probability of unification at AND nodes increases. However, the precise nature of this transition is interesting. Figure 7.9 shows a clear transition between a phase in which almost no answers are inferred to a high inference phase. However, for KB_1, such critical transition was observed only for denser graphs in Model 1 search spaces (i.e., k=7). In other cases, the number of answers inferred remained very low. Figure 7.10 shows similar transition for β=30 and 50 in Model 2 search spaces.

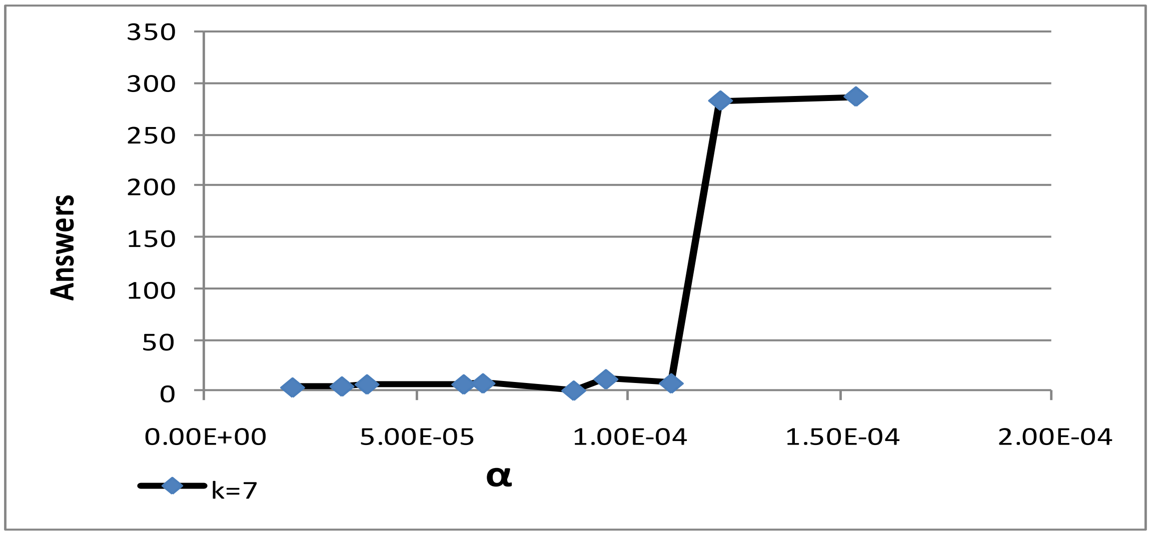

Figure 7.9: Q/A performance for Model 1 search space for KB_1, k=7.

In KB_2, a quick transition from a low inference to high inference is seen in more cases. For Model 1 search spaces, such quick increase in seen when k is 5 or 6 (see Figure 7.11 for the case when k is 6). Similarly, Model 2 search spaces also show two distinct phases of inference (see Figure 7.12 and 7.13). Figure 7.14 shows examples of critical transitions for KB_3.

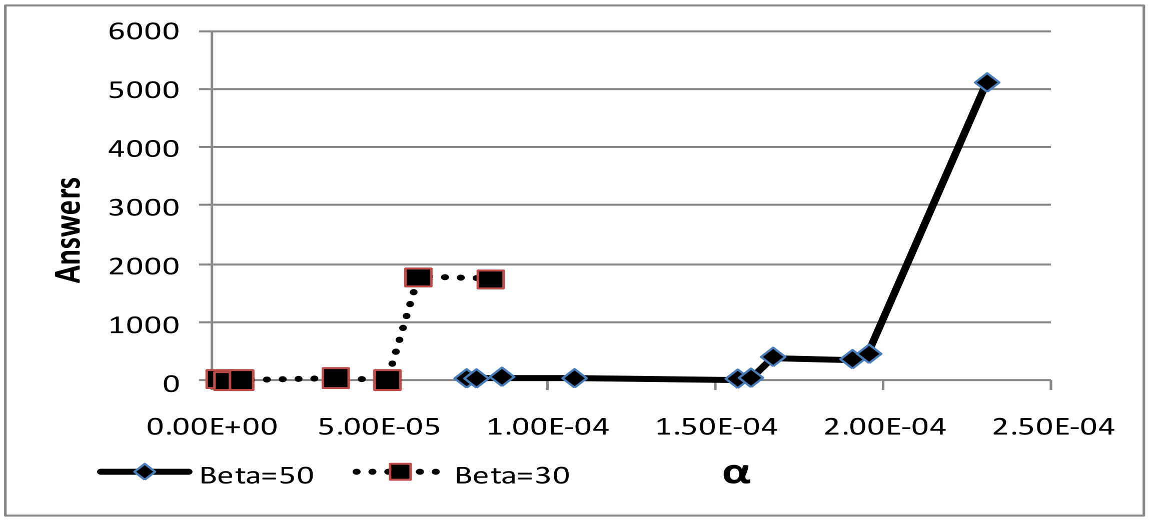

Figure 7.10: Q/A performance for Model 2 search space in KB_1.

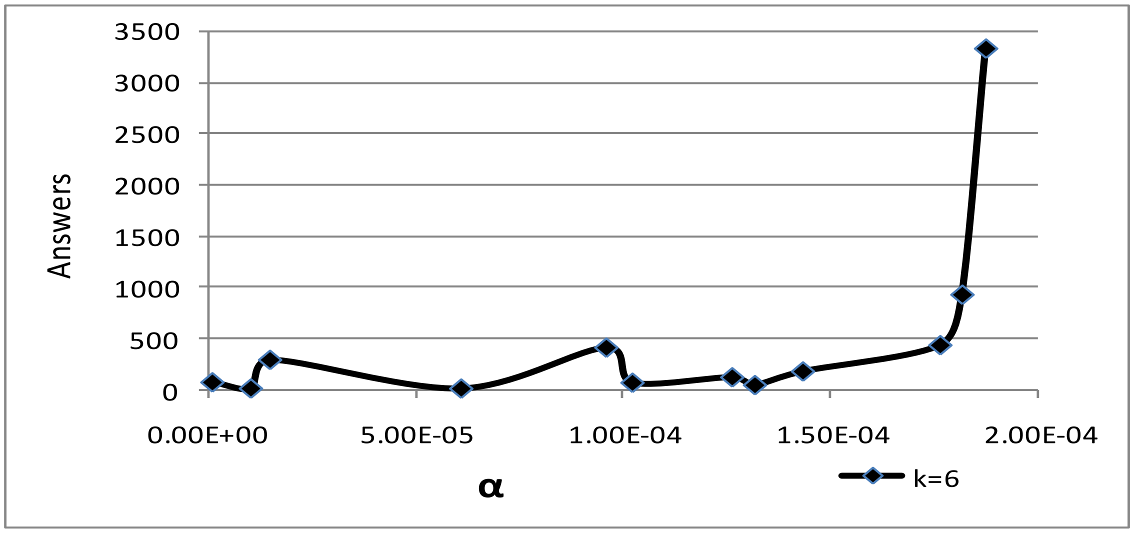

Figure 7.11: Q/A performance for a Model 1 search space in KB_2.

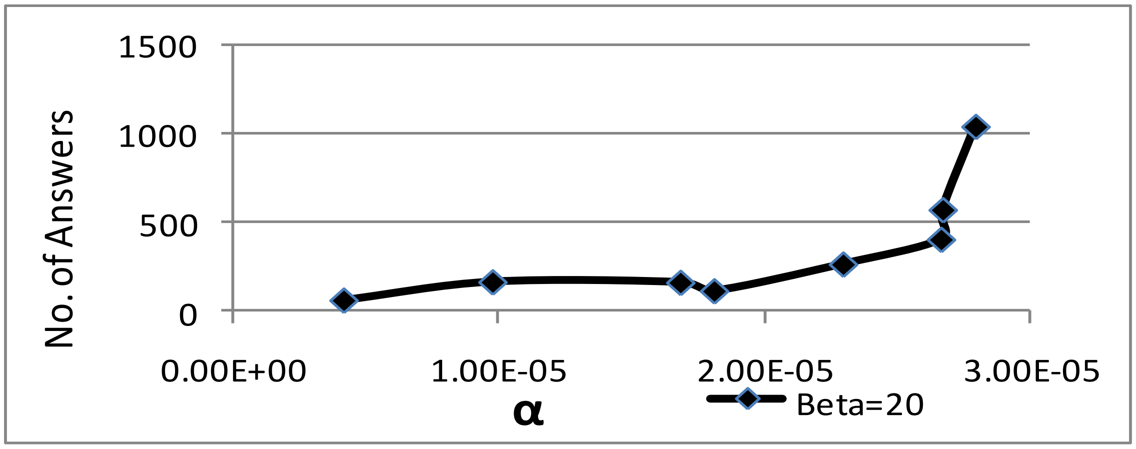

Figure 7.12: Q/A performance for Model 2 search space in KB_2.

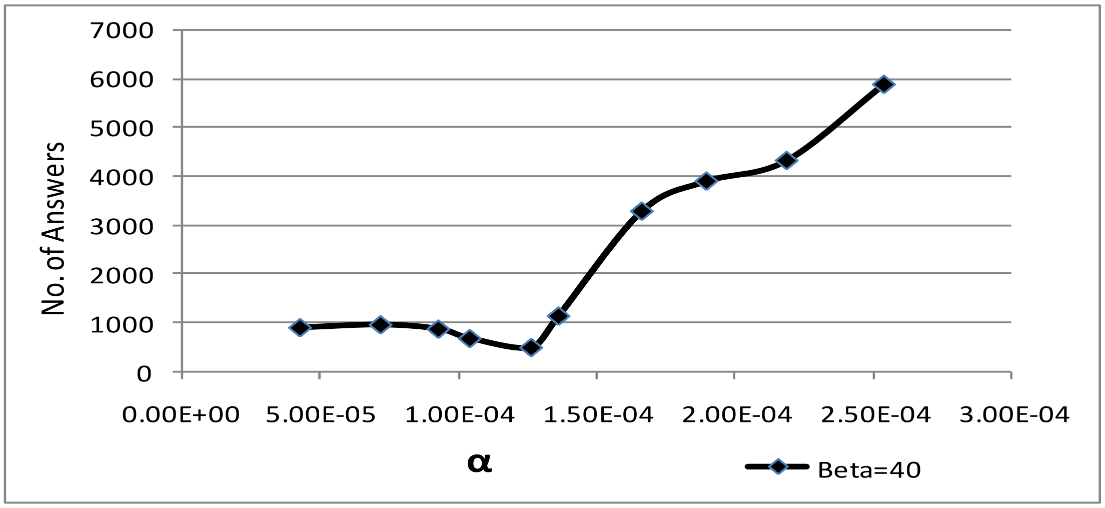

Figure 7.13: Q/A performance for Model 2 search space in KB_2.

In our experiments, about 36% of all search spaces did show a sharp transition discussed above. The results imply that same amount of learning effort can lead to significantly different performance depending on the state of the system. For example, a small addition of facts when the value of α is low will lead to only modest improvement in Q/A performance. On the other hand, when the system is close to a transition point to fast inference, a small amount of learning can provide disproportionate payoff in terms of performance.

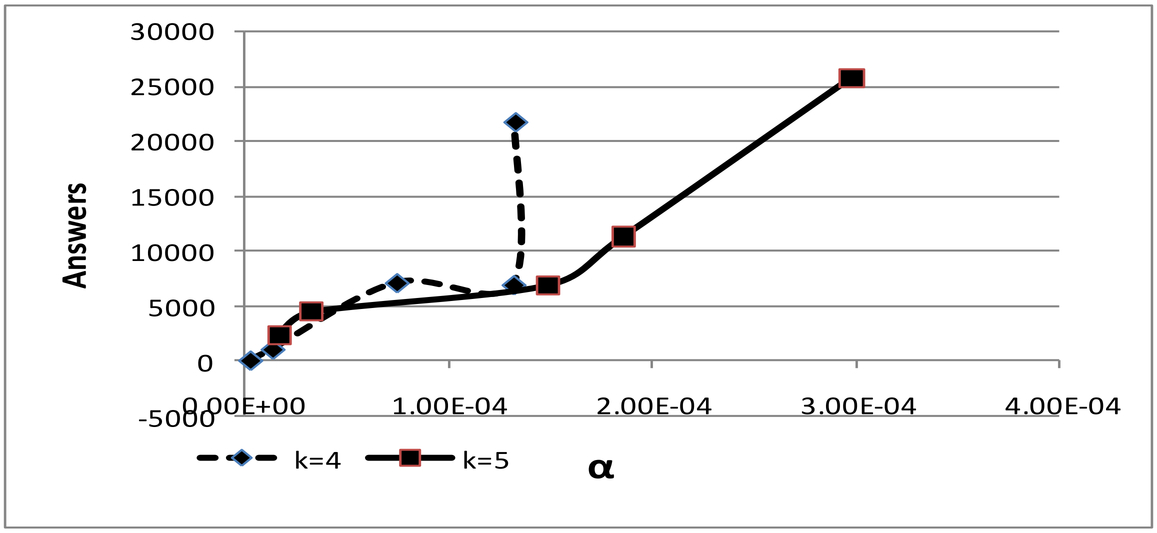

Figure 7.14: Q/A performance for Model 1 search space for KB_3, k=4 and k=5.

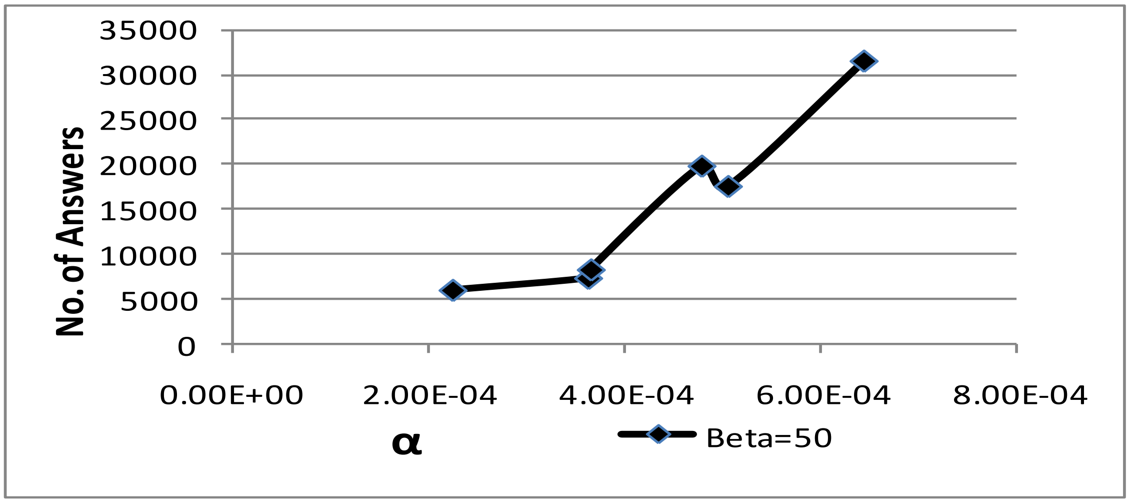

Figure 7.15: No sharp transition for this Model 2 search space in KB_3.

However, in remaining 36% of all cases, no discernible change in the Q/A performance was observed (see Fig 7.15 for an example). In some cases, the spread of inference was limited by the sparseness of the search space. In other cases, larger KBs already showed reasonably high Q/A performance for low values of α, and further improvements were limited by low density of facts.

7.5 Conclusions

As large-scale learning systems mature, there will be a need to steer their learning towards states which lead to progressively better Q/A performance. The study of the structure of search spaces and knowledge, and dynamics of inference are important for attaining this goal. We have proposed and analyzed a model in which simple features of search spaces and knowledge base are used to study the growth patterns of inference. We have reported results for two types of degree distributions and three sizes of KB. The propagation of inference is much less in smaller KBs. Search spaces with uniform degree distributions perform better in larger KBs, whereas relatively skewed degree distributions are more suitable for smaller KBs. Small but critical mismatch between the expectations of axioms and facts in the KB, which lead to almost zero inferences, were observed in 28% of axiom sets generated from the models discussed here. In 36% of all cases, a critical transition between a low-inference to a high-inference region was observed. The methods for selection of axiom sets proposed here (i.e., Model 1 and 2) are fairly general. The low density of facts in ResearchCyc KB hinders the onset of cascades. Next generation learning systems should be cognizant of these properties and the knowledge acquisition cycle should be proactive in guiding the system towards high-inference states. We believe that a meta-reasoning module with the knowledge of these issues can improve the probability of inference cascades. It is hoped that the introduction and study of this model will stimulate further research into understanding the properties of first-order inference and its dependence on the distribution of facts in the KB.

8. A Coordination-based Approach for Focused Learning in Knowledge-Based Systems

8.1 Introduction and Motivation

In recent years, there has been considerable interest in Learning by Reading [Barker et al 2007; Forbus et al 2007, Mulkar et al 2007] and Machine Reading [Etzioni et al 2005; Carlson et al 2010] systems. Rapid progress in these areas has significantly increased the capacity of learning systems to learn new facts from text. Optimal utilization of these facts could change the face of modern AI systems. However, we must make sure that the benefits of these new facts show up in better Q/A performance. Inundating a knowledge-base (KB) with irrelevant facts is hardly useful. Therefore, a rational learning system should try to formulate small number of learning goals which would help it to answer more questions. Which learning goals should be selected to maximize Q/A performance?

Techniques for selecting a small number of queries are also needed for active learning systems, which interact with a human expert or crowds to augment their knowledge. Since it would be impractical to seek thousands of facts from a single human user, learning systems must limit the scope of their queries. Even with crowdsourcing, selecting queries that avoid gathering irrelevant information is important.

In this chapter, we argue that an arbitrary selection of queries would result in large scale unification problems and the effect of new facts would not reach the target queries. We show that the selection of queries for maximizing Q/A performance is similar to a coordination game. We then use reinforcement learning to solve this problem. We model the dynamics of a learning system which sends learning goals to an external knowledge source, and experiments show that this coordination-based approach helps in improving Q/A performance. The results entail that the dependencies of search space induce a small partition (for each query) of the entire domain, which is selected by the reinforcement learning algorithm.

The rest of this chapter is organized as follows. We start by discussing relevant work. We then discuss our learning model and its similarities to coordination games. We conclude after describing experimental results.

8.2 Related Work

Our work is basically inspired by the idea that the deductive query graphs should be seen as a network of agents and coordination between them is needed for optimal results. The formal study of coordination in a network of agents has been done in social sciences [Jackson 2008], and mainstream computer science [Kleinberg & Raghavan 2005]. In [Kleinberg & Raghavan 2005], the authors have studied how game-theoretic ideas can be used for allocation of resources in query networks. In the AI literature, there has been some tangentially related work. For example, the relation between game theory and Boolean/first-order logic has been studied [Tang & Lin 2009, Dunne et al 2008]. Researchers in database community have discussed the importance of combining different constraints for optimal query plans [Hsu & Knoblock 2000]. QSAT problems have been seen as a two-person game [Kleinberg & Tardos 2005]. Although we have benefitted from these endeavors, we are not aware of any prior work which has studied the similarity of coordination games and the dynamics of knowledge-based learning systems.

8.3 Background: Coordination Games

Game theory has been extensively used for the study of interaction among independent, self-interested agents. The normal-form representation of a game is widely used to describe a game [Shoham & Leyton-Brown 2009]:

Definition 8.1 (Normal-form game): A (finite, n-person) normal-form game is a tuple (N, A, u), where:

- N is a finite set of n-players, indexed by i;
- $A = A_1 \times \ldots \times A_n$, where A_i is a finite set of actions available to player i. Each vector $a = (a_1, \ldots, a_n)$ is also called an action profile.
- $u = (u_1, \ldots, u_n)$, where $u_i : A \rightarrow R$ is a real-valued utility (or payoff) function for player i.

Coordination games are a restricted type of game in which the agents have no conflicting interests. They can maximize benefits for all agents by coordinating their actions.

Example: A classic example of coordination game is the so-called Battle of Sexes[37]. In this game, a couple has decided to meet this evening but cannot recall if they will be attending the opera or the football match. The husband prefers the football match, whereas the wife prefers the opera. However, both would like to go to the same place rather than different ones. If they cannot communicate, where should they go? The payoff matrix for this game is shown below. Here the wife chooses a row and the husband chooses a column. In each cell, the first and the second number represent the payoff to the wife and the husband respectively.

		Husband	
		Opera	Football
	Opera	3,2	0,0
Wife	Football	0,0	2,3

Table 8.1: The Battle of Sexes Payoff Matrix

8.4 A Model for Learning Systems

Progress in deductive Q/A performance will require reasoning effectively with a continuously growing KB and 10^3-10^6 first-order axioms. It is obvious that we would like to choose only those facts which are relevant for answering questions. Acquiring many irrelevant facts is inadvisable for two reasons: (a) Acquiring these facts from a human expert is expensive, and even crowdsourcing is not free and (b) These facts could affect the performance of deductive reasoning by increasing the number of failed reasoning chains. Therefore, large knowledge-based learning systems should prefer learning goals that ensure that they get only the most relevant facts from the external knowledge source. Very general learning queries

[37] A discussion of this problem is available in any standard game-theory textbook. This example is from Wikipedia.

like `(<predicate> ?x ?y)` would result in acquisition of large number of facts . Therefore, in this work we use a learning model in which the learning queries are of the type $(p_i <C_j> <C_k>)$, where C_j and C_k are specific collections[38]. In what follows, we will discuss how the set of axioms used by the Q/A system plays an important role in determining the learning goals.

A high-level view of our model of learning system is shown in Figure 8.1. To simulate the learning behavior, we are using the inverse ablation model described in [Sharma & Forbus 2010]. The basic idea is to take the contents of a large knowledge-base (here, ResearchCyc) and make a simulation of the learning system by removing most of the facts. The operation of the learning component is simulated by gathering facts from the ablated portion of the KB that satisfy the learning requests, and adding them to the test KB. The initial KB consists of the basic ontology definitions (i.e., `BaseKB` and `UniversalVocabularyMt`) plus 5,180 facts chosen at random. The rest of the ResearchCyc KB acts like an external knowledge source. At every time *t*, the learning system sends queries of the type $(p_i <C_j> <C_k>)$ to an external knowledge source. The answers are stored in KB(t) to get KB(t+1). For example, an example of such a learning request could be `(doneBy <Buying> <BritishCorporation>)`[39]. Which queries should be sent to the external knowledge source? Our aim is to find the values of C_js, which would lead to acquisition of those facts from the external knowledge source which would maximize the Q/A performance.

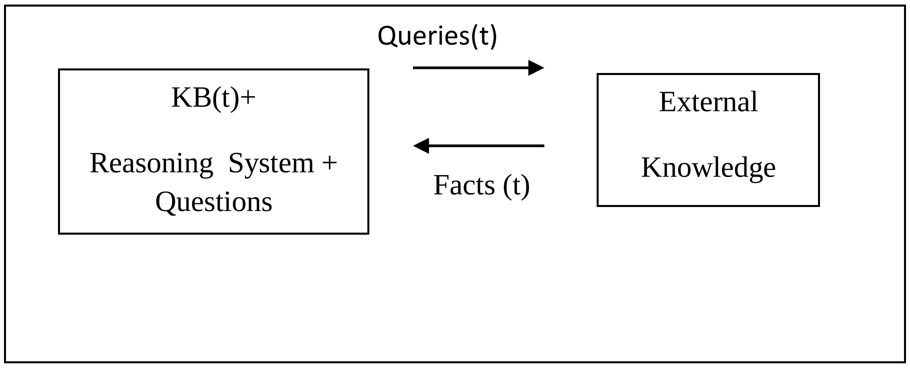

Figure 8.1: An idealized model of learning systems

[38] The set of specific collections consists of all collections which have less than N instances. In this work, N was set to 5000.
[39] This query represents all '`doneBy`' statements involving buying actions done by British corporations.

Assume, for example, that we expect the learning system to find the country in which a person was born. In Figure 8.2, we show a simplified version of the search space which infers the country in which a person was born. The rectangular boxes represent AND nodes[40]. Note that the root node has been simplified in Figure 2; it actually refers to the formula `(holdsIn ?Birth-Event (objectFoundInLocation ?Person ?Location))`. For example, we may be expected to answer this query for the following set of people:

Q = {Einstein, Fermi, Riemann, Laplace, Hitler, Mao, Gauss, Feynman, Oppenheimer}

The variables in other nodes have not been shown for simplicity. To answer these queries and maximize the influence of axioms, the learning system needs facts involving the leaf nodes of the search space. Let us assume that the learning system decides to send following queries to the external knowledge source[41]:

t=0: `(geographicalSubRegionsOfState <C1> <C2>).`
t=1: `(geoPoliticalSubDivisions <C3> <C4>)`
t=2: `(birthChild <C5> <C6>)`

The aim of the learning system is to select the values of C1,…,C6 such that the Q/A performance is maximized. Let us consider two scenarios:

Scenario 1: C1= `US-State`, C2= `USCity`, C3=`AfricanCountry`, C4=`AfricanCity`, C5= `BirthEvent`, C6= `FrenchPhysicist`.

Scenario 2: C1= `US-State`, C2= `USCity`, C3=`US-State`, C4=`USCity`, C5= `BirthEvent`, C6= `USPhysicist`.

[40] We are assuming an AND/OR query graph, where unification of antecedents is needed at AND nodes.
[41] We have ignored the `eventOccursAt` leaf node for simplicity.

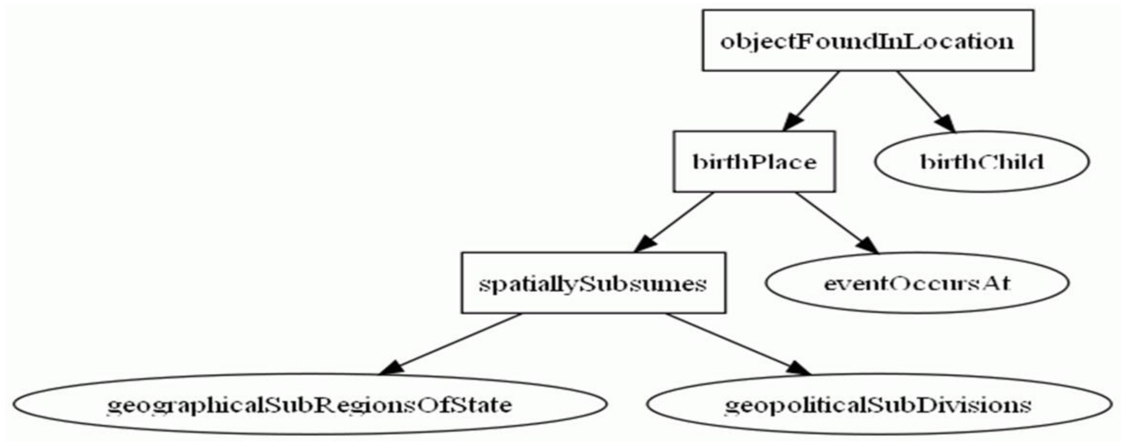

Figure 8.2: Simplified view of search space (Square nodes represent AND Nodes, Oval nodes represent OR nodes)

Let KB(0) represent the contents of the initial KB. Let Facts_1 and Facts_2 represent the set of facts acquired from the external knowledge source in scenarios 1 and 2 respectively. Then KB(0) ∪ Facts_1 would lead to little or no unification at the `spatiallySubSumes` and `objectFoundInLocation` nodes. Very few answers would be inferred at the root node in this case. On the other hand, unification problems would not arise for the KB consisting KB(0) ∪ Facts_2, and a significant number of queries would be answered.

The left child of the `objectFoundInLocation` node in Figure 2 infers the location of birth of the person, whereas its right child encodes the ethnicity of the person. As discussed above, lack of synchronization between what is inferred by different regions of the search space could lead to failures in inference. Let `?Location` and `?Ethnicity` be the variables which refer to what is inferred by the left and right child of the root node respectively. For simplicity, let us assume that the values of `?Location` and `?Ethnicity` are limited to French/American cities and scientists. Table 2 summarizes the relation between Q/A performance and the choice of values for the variables. We note that this explanation has been simplified in three ways: (a) The domains for these variables are much bigger, (b) Such tables can be made for every AND node in the search space, and (c) If we consider a bigger domain, we will see that the change in Q/A performance is not as abrupt as shown in Table 8.2, but we would observe a more gradual and graded change as the synchronization between different search branches improves.

	?Location= < FrenchCity>	?Location= <State-US>
?Ethnicity = <FrenchPhysicist>	42	0
?Ethnicity = <USPhysicist>	0	58

Table 8.2: Relation between choice of learning queries and Q/A performance

Notice the similarity between Table 8.1 and 8.2. In both cases, the top left and bottom right cells have significantly higher payoffs than other two cells. This variance arises due to the different expectations from different regions of the search space. In the example discussed above, we saw the possibility of spatial inconsistency between the outputs of different inference branches. Other types of inconsistencies (e.g., temporal inconsistencies) are also possible.

Now we can define the problem of coordination of learning actions in knowledge-based systems, to make the correspondences clear:

Definition 8.2: A normal-form game in a first-order learning system is a tuple (*N*, *A*, *u*), where:

- *N* is a finite set of variables, indexed by *i*;
- $A = A_1 \times \ldots \times A_n$, where A_i is a finite set of domains available to variable *i*.
- $u=(u_1, \ldots, u_n)$, where $u_i: A \rightarrow R$ is a real-valued utility (or payoff) function for player *i*.

We are assuming that the variable names are unique. In fact, there should be one agent/player for each argument position of predicate. For example, if `(objectFoundInLocation ?person ?location)` and `(cityInCountry ?city ?country)` are two nodes in the search space then:

N = `{?person, ?location, ?city, ?country}`

A_1= {`FrenchPerson,ChinesePerson,MuslimPerson`,. }

A_2= {`EuropeanCountry,US-State, AfricanCountry`, …}

A_3= {USCity, AfricanCity, BritishCity,…}

A_4= {AsianCountry, EuropeanCountry, …,…}

The domains for A_is should be filtered to ensure that very general collections are excluded. The utility function maps the learning actions to the number of questions answered from facts acquired from their execution. Given this definition, we can say that the choice of optimal selection of learning queries in knowledge-based systems is similar to synchronization of actions in a coordination game.

Can the problem of synchronization be avoided if our learning requests are not of the type (predicate <*Collection*> <*Collection*>)? We do not believe so. Recall that the problem arises due to two reasons: (a) From our inability to get all relevant facts from a human expert, and (b) Increase in the number of failed reasoning chains due to irrelevant facts. These problems would continue to arise even if we use an entity-based learning strategy[42]. For example, a Machine Reading or learning by reading system can easily gather thousands of facts about topics like UnitedStatesOfAmerica from the Web. We cannot guarantee that all these facts would be useful because the utility of facts is determined by the set of axioms and their expectations. This implies that getting a smaller set of facts about more specific sub-topics (e.g., "Foreign Policy of United States") could lead to better and more efficient Q/A performance if they combine well with other queries. Similarly, it would be better to design specific and synchronized queries about entities for a human expert than to acquire small number of disconnected facts about a bigger topic.

Can we avoid this problem by using a centralized topic selection approach? Note that there is no simple mapping between the topic to the optimal set of learning queries. For example, if the learning system chooses to learn about the US economy, there is no obvious method which could tell us which aspect of this complex problem would be most useful for answering questions[43]. In fact, the optimal selection of actions in the coordination game discussed above maps the topic to the best set of queries.

This formulation implies that learning systems will need to find the optimal values of C_is to maximize Q/A performance. When we choose values for these variables, we choose to reason about a small context

[42] In an entity-based learning strategy, the learning system seeks facts about entities like UnitedStatesOfAmerica or BillClinton.

[43] In particular, we could seek facts about (i) China's currency policy, (ii) Agricultural sector in the US, and (iii) Japan's comparative advantage in electronics goods. Which of these approaches would lead to best Q/A performance?

or a partition (e.g., Location of French Physicists, US Foreign Policy in Europe, etc.) of the entire domain.

8.5 Reinforcement Learning

Reinforcement learning has been used for solving many problems, including game-theory. Our approach is based on the algorithm discussed in [Claus & Boutilier 1998, Bowling &Veloso 2002]. Their algorithm is based on using a Joint Action Learner (JAL), which is aware of the existence of other agents and uses reinforcement learning to learn how different combination of actions affect the performance. Agents repeatedly play a *stage game* in which they select an individual action and observe the choice of other agents. The net reward from the joint action is observed. This basically means that the environment of the repeated game is stationary (i.e., independent of time and history). However, the actions chosen by the agents are not independent of time and history. In fact, our aim is to show that *repeated games* would help the learning system to become cognizant of the interdependencies in the search space and guide the learning system to seek the optimal set of facts from the external knowledge source. The algorithm is shown in Figure 8.3.

The initial state of the system, s_0, is the initial state of the inverse ablation model (i.e., Ontology plus 5180 facts chosen at random). If the number of agents is k, and the size of each A_i is n, then the total number of actions is n^k. Therefore, the set of all states is $\{s_0, s_1, s_2, \ldots, s_{nk}\}$.

Recall that we have an agent for each argument position of every predicate, leading to a set of N agents and each agent i can choose from a finite set of individual actions A_i. The chosen actions at any instance of the game are joint actions. The set of joint actions is $A_1 \times A_2 \times \ldots A_n$. The notation A_{-i} refers to the set of joint actions of all agents excluding agent i and a_i refers to the action selected by agent i. In step 2b of the algorithm, the algorithm keeps a count of the number of times an agent has used a given action in the past. These relative frequencies provide information about the agent's perception of the worth of different actions, and help the algorithm to have a model of the agent's current strategy. Each agent then uses them to choose a best response for the strategies of others [Claus & Boutilier 1998]. In step 2a of Figure 8.3, $C(s, a_{-i})/n(s)$ is the estimate that other agents would select the joint action a_{-i} based on the history of play and their current strategy. Therefore, we choose action a_i which would be the optimal response to this joint action.

Algorithm: Learning for player i.

```
1. Initialize Q arbitrarily, and for all $s \varepsilon S$, $a_{-i}$ $\varepsilon A_{-i}$, set
      a. $C(s, a_{-i}) \leftarrow 0$, $n(s) \leftarrow 0$
2. Repeat,
      a. From state s select action $a_i$ that maximizes, $\sum_{a_{-i}} \frac{R}{n(s)}$,
         where, $R = C(s, a_{-i})*Q(s, <a_i, a_{-i}>)$.

      b. Observing other agents' actions $a_{-i}$, reward $r$, and next state $s'$, set
            $Q(s, a) \leftarrow (1-\alpha)\, Q(s, a) + \alpha(r + \gamma V(s'))$
            $C(s, a_{-i}) \leftarrow C(s, a_{-i}) +1$
            $n(s) \leftarrow n(s) +1$

         where,
               $a= (a_i, a_{-i})$
           $V(s) = \max a_i \sum_{a_{-i}} \frac{R}{n(s)}$.
```

Figure 8.3: Joint-Action Learning Algorithm [Bowling &Veloso 2002, Claus & Boutilier 1998]

8.6 Experimental Analysis

Learning by Reading systems typically use a Q/A system to use what the system has learned. For example, Learning Reader used a parameterized question template scheme [Cohen *et al*, 1998] to ask questions. In this work, we have evaluated the efficacy of algorithm on five types of questions used in Learning Reader. These question templates were: (1) Where did <*Event*> occur?, (2) Who was the actor of <*Event*>?, (3) Where might <*Person*> be?, (4) Who was affected by the <*Event*>?, (5) Where is <*GeographicalRegion*>? These questions were selected because the KB mainly contains information relevant for answering these questions. To test the performance for other predicates we also included a sixth query template `(temporallyIntersects ?x ?y)`. Since `temporallyIntersects` is a very general predicate, 2,038 other specializations are accessible through backchaining. In each template, the parameter (e.g., <*Person*>) indicates the kind of thing for which the question makes sense (specifically, a collection in the Cyc ontology). We use these questions in our experiments below, to provide realistic test of reasoning. When answering a parameterized question, each template expands into a set of formal queries, all of which are attempted in order to answer the original question. Each template contains one open variable, whose binding constitutes the answer. Our FIRE reasoning system uses backchaining over Horn clauses with an LTMS [Forbus & de Kleer 93]. We limit inference to Horn clauses for tractability. We use network-based optimization techniques for automatically selecting an efficient set of axioms. Inference is limited to depth 5 for all queries, with a timeout of 90 seconds per query.

In this chapter we have argued that learning systems would be able to improve their Q/A performance by using coordination algorithms. Therefore, for the baseline, we are using an approach which is oblivious of the presence of other agents/nodes. In other words, the baseline algorithm chooses the learning goal which had yielded the highest number of facts in the past (plus some exploration). We compare the performance of this baseline with the coordination algorithm shown in Figure 8.3 which is cognizant of the presence of other nodes and tries to learn the effect of dependencies in the search space. To ensure adequate exploration, the agents chose a random action with probability 0.05. The learning rate, α, was set to 0.5. We report results for the six question types mentioned above. The results are shown below.

Exp No.	Algorithm	No. of Queries	No. of Answers	Improvement w.r.t. Baseline
1	Baseline	13,153	4,878	-
	Coordination	13,153	8,487	74%
2	Baseline	13,153	1,104	-
	Coordination	13,153	1,556	41%
3	Baseline	5,299	249	-
	Coordination	5,299	315	26%
4	Baseline	13,153	510	-
	Coordination	13,153	627	23%
5	Baseline	33,915	4,856	-
	Coordination	33,915	12,233	151%
6	Baseline	36,564	9,211	-
	Coordination	36,564	15,658	70%

Table 8.3: Experimental Results

The method proposed performs better for Experiments 1, 5 and 6 because the density of facts for these questions types was higher. In such cases, a coordination-based approach is more likely to be useful due to higher probability of unification. It is clear that it will be difficult to find the optimal solution if the domains, i.e. A_is, are large. To understand how the size of the search space affects the performance, we plotted the maximum number of answers derived when all elements of A_is with fewer instances than a certain threshold were rejected. As we lower this threshold, more and more collections are chosen. Figure 8.4 shows the effect of this threshold on the performance. A threshold of -1 indicates that all collections are selected. We see that as we increase the threshold from -1 to 10, performance improves. This implies that the optimal solution couldn't be found due to the large search space when all collections were included. However, when we continue to increase the threshold, we find that the performance deteriorates. This happens because we have removed too many collections and optimal solution cannot be found from the smaller set of collections.

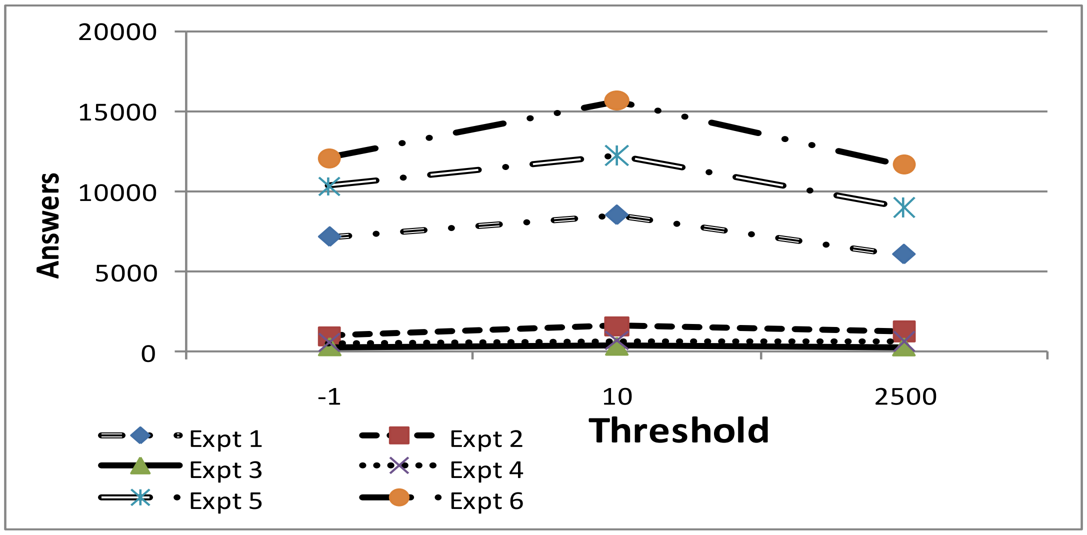

Figure 8.4: Effect of threshold on performance

8.7 Conclusion

Large knowledge-based systems often need to identify a set of learning goals which could be answered with the help of an external knowledge source. We have shown that this problem is similar to a coordination game and reinforcement learning can be used for solving this problem. This approach finds a small partition which is induced by different expectations from different branches of the search space. Queries from this partition become learning goals. Experiments show that this coordination-based approach helps in improving Q/A performance. These results suggest three lines of future work. First, due to semi-decidable nature of first-order reasoning with Horn axioms, it is difficult to determine if we have achieved the best possible performance. Therefore, we would like to see if an algorithm like WoLF[Bowling & Veloso 2002] can further improve the performance. Secondly, we have noticed huge variance in the final solution even when values of most variables are fixed. This shows that some variables have more pivotal position and wield more influence in the search space. We would like to find if structures like "backbones" and "backdoors" (as discussed in SAT literature) can be identified here. Finally, we plan to implement these techniques in a real learning system and study its performance.

9. Conclusion and Future Work

In this work, we have suggested some methods for improving Q/A performance in large knowledge-based systems. We have also proposed a model for studying the evolution of learning systems.

In chapter 3, we suggested that systemic properties of KB play an important role in determining the hardness of problems. We proposed two heuristics for simplifying the search space. Results show that such methods are useful for making reasoning more efficient.

In chapter 4, we observed that the contents of KB can be analyzed to calculate likelihoods of success of queries. We found that due to non-uniform distribution of facts in KB, many queries are unlikely to succeed. Moreover, the constraints in antecedents of axioms impose very strong constraints. This ensures that the consequent of such axiom is inferred in very few cases. We identify and prune such queries and axioms. Results show that such an approach is useful for improving Q/A performance.

We addressed the problem of knowledge gaps in chapter 5. We argued that a graph-based path-finding algorithm should be used to find plausible inference chains in knowledge-based systems. Knowledge patterns written in terms of predicate types were used to guide and limit path exploration. Reinforcement learning was used to learn the plausibility of these patterns.

In chapter 6, we proposed a model for understanding the evolution of knowledge-based systems. We studied the trends of some systemic properties of KB for two learning strategies. We found significant differences in their properties. These strategies were not able to learn more than 33% of maximum possible facts. We concluded that we will need a set of learning strategies starting from different starting points to improve the coverage.

In chapter 7, we studied the growth of Q/A performance in an inverse ablation-based framework. We found that in some cases, a critical transition was seen from a low inference state to a high inference state. We believe that meta-reasoning modules should be cognizant of these issues and guide learning systems to such high inference states.

The importance of coordination of learning actions was discussed in chapter 8. Dependencies in search spaces ensure that not all facts are reasonable learning goals. We showed that there are similarities between choosing a set of learning goals and solving a coordination game. Reinforcement learning was then used to solve this problem.

In what follows, we discuss some possible improvements to the methods and models discussed in previous chapters.

- In [Horvitz & Klein 1995], the authors present a Bayesian analysis of the truth of a propositional claim, based on the size of search space and progress towards a goal during theorem proving. Extending this work for first-order logic would be very useful for inference engines.
- The concept of backdoors has received attention in the propositional SAT community. Given a SAT formula, a set of variables forms a *backdoor* for a problem instance if there is a truth assignment to these variables such that the simplified formula can be solved in a polynomial time via normal SAT solvers [Ruan et al 2004]. Understanding whether such structures exist in large commonsense KBs would be useful. There are, of course, many differences between propositional SAT solving and Q/A in first-order KBs. SAT solvers easily find inconsistencies in propositional problems. On the other hand, as far as the current version of ResearchCyc KB is concerned, it is fairly difficult to find inconsistencies during the inference process. Therefore, exact correspondences between these domains wouldn't be found. However, it is possible that some variables in first-order inference problems hold a more pivotal position than others. Assigning values to those variables could simplify the rest of the problem.
- There has been considerable work in the constraint satisfaction community which can be used for optimizing first-order reasoning in large KBs. For example, the importance of ordering variables and their significance for backtrack-free search has been recognized [Dechter 2003]. We might be able to use those insights to understand how partial solutions in first-order logic should be represented and extended without backtracking. We believe that such techniques would lead to significant improvements in reasoning performance. Similarly, there has been some work on learning while solving constraint satisfaction problems [Dechter 2003]. Implementing a module which could improve its performance with experience would be useful.
- The work on axiom extraction (see Chapter 3) helps us to extract an efficient set of axioms. In future, we should study how the algorithm would perform when a KB is growing rapidly. Similarly, we would like to study how the work on static analysis would be affected by a rapidly changing KB.

- We used some systemic properties of search space to study the growth patterns of inference in chapter 7. This approach has the problem that it mostly ignores the fact that inference propagates from ground facts from the leaves to the root node in the search space. This “directional” nature should be taken into account. Recently, some researchers have used ideas from control systems to study the controllability of complex networks [Liu et al 2011]. We believe that these ideas could be useful for improving our model.
- We believe that ideas from operations research (OR) might be useful for solving some problems in knowledge-based systems. For example, identifying an efficient set of axioms is an important task in our context. This task should include creation of some new nodes in the search space. In OR, the network synthesis problem selects some edges from a given set. If an arc (i, j) is constructed with capacity u_{ij}, then a corresponding cost of $c_{ij}>0$ per unit capacity will be incurred. The constructed network must independently sustain a certain maximal flow requirement $r_{ij}>0$ between each pair of nodes or terminals [Bazaraa et al 2005]. The aim is to select a minimal cost network, which satisfies these flow requirements. This network synthesis problem is similar to the problem of choosing an efficient set of axioms. Note that if we start with a complete graph, then the network synthesis problem would choose best paths from the leaves to the root. In our context, the “flow” should be seen as the flow of inference from one node to another.
- The problem of knowledge gaps is important for large knowledge-based systems. In [Crandall et al 2010], the authors discuss the extent to which social ties between people can be inferred from co-occurrence in time and space. We believe that missing facts in KBs can be found using spatial and temporal co-occurrences in ResearchCyc KB. However, we will need to solve at least two problems:
    - The ResearchCyc KB does not have enough information about the size of spatial entities. Since the probability of existence of a link decreases with the size of spatial cell, such information is needed for understanding the likelihood of a link.
    - Secondly, spatial and temporal co-occurrences can only help in predicting the existence of a link. We will need to use other methods to determine the precise nature of relation between the entities.

10. References

E. Amir and S. McIlraith. Partition-Based Logical Reasoning for First-Order and Propositional Theories, *Artificial Intelligence*, 162(1-2), pages 49-98, 2005.

K Barker, B Agashe, S Y Chaw, J Fan, N. Friedland, M Glass, J. Hobbs, E. Hovy, D. Israel, D Kim, R Mulkar-Mehta, S Patwardhan, B Porter, D Tecuci, P Z. Yeh: Learning by Reading: A Prototype System, Performance Baseline and Lessons Learned. *Proc. of AAAI* 2007: 280-286

C. Baral. Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003.

M. S. Bazaraa, J. Jarvis and H. Sherali. Linear Programming and Network Flows. Wiley Interscience, 2005

M. Belduccinni, C. Baral and Y. Lierler (2008). Knowledge Representation and Question Answering. In Vladimir Lifschitz and Frank van Harmelen and Bruce Porter, ed In Handbook of Knowledge Representation.

M. Bowling and M. Veloso. Multiagent Learning Using a Variable Learning Rate. *Artif. Intell.*, 136, pp. 215-250, 2002

A. Braunstein, M. Mezard, and R. Zecchina (2005). Survey propagation: An algorithm for Satisfiability, *Random Structures and Algorithms*, 27, pp. 201-226

R. J. Brachman and H. J. Levesque, Knowledge Representation and Reasoning, Morgan-Kaufmann, 2004

E. Brill, S. Dumais and M. Banko (2002). An analysis of the AskMSR question-answering system. *Proc. of ACL*, 2002

E. Brill, S. Dumais and M. Banko (2002). An analysis of the AskMSR question-answering system. *Proc. of ACL*, 2002.

A. Carlson, Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. and Mitchell, T. 2010. Toward an architecture for Never-Ending Language Learning. *Proceedings of AAAI*, 2010.

D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec and C. Faloutsos. Epidemic Thresholds in Real Networks. *ACM Trans. On Information and System Security*, 10(4), 2008

E. Charniak. A Neat Theory of Marker Passing. *Proc of AAAI*, 1986

P. Clark, J. Thompson and B. Porter (2000). Knowledge Patterns. *Proc. of KR*, 2000

C. Claus and C. Boutilier. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems.*Proc. of AAAI*, 1998

P. Cohen, Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., and Burke, M. 1998. The DARPA High-Performance Knowledge Bases Project. AI Magazine, 19(4), Winter, 1998, 25-49

A. Collins. (1978). Human Plausible Reasoning. BBN Report No. 3810

D. Crandall, L. Backstrom, D. Cosley, S. Suri, D. Huttenlocher, J. Kleinberg. Inferring Social Ties from Geographic Coincidences. *Proc. National Academy of Sciences* 107(52) 22436-22441, 2010.

R. Dechter. Constraint Processing, Morgan Kaufmann, 2003.

P. Domingos and M. Richardson. Mining the Network Values of Customers. *Proc. of Conf. on Knowledge Disc. and Mining*, 2001.

P. Dunne, W. der Hoek, S. Kraus and M. Wooldrige. Cooperative Boolean Games. *Proc. of AAMAS*, 2008.

S. Dzeroski, L. de Raedt and K. Driessens (2001). Relational Reinforcement Learning. *Machine Learning*, 43, pp. 7-52

O Etzioni, M J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, Alexander Yates: Unsupervised named-entity extraction from the Web: An experimental study. *Artif. Intell.* 165(1): 91-134 (2005)

C. Faloutsos, K. S. McCurley and A. Tomkins (2004). Fast Discovery of Connection Subgraphs. *Proc. of KDD*, 2004

D. Ferrucci, E. Nyberg *et al* (2009). Towards the Open Advancement of Question Answering Systems. IBM Research Report. RC24789 (W0904-093), IBM Research, New York

K. D. Forbus and J. de Kleer. *Building Problem Solvers*. MIT Press, 1993

K D. Forbus, C Riesbeck, L Birnbaum, K Livingston, A Sharma, L Ureel II: Integrating Natural Language, Knowledge Representation and Reasoning, and Analogical Processing to Learn by Reading. *Proc . of AAAI* 2007: 1542-1547

A. M. Frisch, Knowledge Retrieval as Specialized Inference. Ph. D. Thesis. University of Rochester, 1987.

A. Giordana and L. Saitta. Phase Transitions in Relational Learning. *Machine Learning,* 41, pp. 217-251, 2000.

Greiner, R. and Orponen, P. (1991). Probably Approximately Optimal Derivation Strategies. *Proc. of Second Intl. Conf. On Knowledge Representation and Reasoning,* Boston, May 1991.

E. Grois and D. Wilkins (2005). Learning Strategies for open-domain natural language question answering. *Proc. of IJCAI,* 2005

C. Gomes, Fernandez, B. Selman and C. Bessiere. Statistical Regimes Across Constrainedness Regions, *Proc.of CP*, 2004.

E. Horvitz. (1990). Computation and Action under Bounded Resaources. Ph. D Thesis. Stanford University.

E. Horvitz and A. Klein (1995). Reasoning, Metareasoning, and Mathematical Truth: Studies of Theorem Proving Under Limited Resources. *Proc. of Conf. on Uncertainty in Artificial Intelligence*, 1995.

C. Hsu and C. A. Knoblock. Semantic Query Optimization for Query Plans of Heterogeneous Multidatabase Systems. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 6, 2000.

M . Jackson. Social and Economic Networks. Princeton University Press, 2008

H. Jeong, S. P. Mason, A. L. Barabasi and Z.N. Oltavi. Lethality and Centrality in Protein Networks, *Nature*, 411, pages 41-42, May 2001

L. P. Kaelbling, M. L. Littman and A. W. Moore. (1996). Reinforcement Learning: A Survey. *Jl. of AI Research*, 4, pp. 237-285, 1996

H. Kautz and B. Selman. The state of SAT. *Discrete Applied Mathematics*. 155(12), pp. 1514-1524, 2007

R. Khardon. (1999) Learning Function-Free Horn Expressions. *Machine Learning*, 37, pp. 241-275

P. Kilby, J. Slaney, S. Thiebaux and T. Walsh, Backbones and Backdoors in Satisfiability, *Proc. of AAAI*, 2005

J. Kleinberg and P. Raghavan. Query Incentive Networks. *Proc. of 46th IEEE Symp. on Foundations of Computer Science*, 2005

J. Kleinberg. Navigation in a small world. *Nature*, 406, page 845, 2000.

J. Kleinberg and E. Tardos. Algorithm Design. Addison Wesley, 2005

G. Kossinets and D. Watts. Empirical Analysis of an Evolving Social Network. *Science*, 311, 88, 2006

R. Kowalski. A Proof Procedure Using Connection Graphs. Journal of ACM, 22 (4), pp. 572-595, 1975

J. Lang, Liberatore, P. and Marquis, P. (2003). Propositional Independence: Formula-Variable Independence and Forgetting, *Journal of Artificial Intelligence Research*, 18, pp. 391-443

Ledeniov, O. and Markovitch, S. (1998). The Divide-and-Conquer Subgoal-Ordering Algorithm for Speeding up Logic Inference. Journal of Artificial Intelligence Research, 9, pp. 37-97

D. Lenat. The Dimensions of Context-Space. Available at http://www.cyc.com/doc/context-space.pdf, 1998

D. B. Lenat and R. Guha, Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project, Addison Wesley, 1989.

J. Leskovec, J. Kleinberg and C. Faloutsos. Graph Evolution: Densification and Shrinking Diameters, *ACM Trans. on Knowledge Discovery from Data*, Vol 1, No. 1, 2007.

E. Lieberman, C. Hauert and M. A. Nowak, Evolutionary Dynamics on Graphs, *Nature*, 433, pp. 312-316, 2005

Y. Liu, J. Slotine and A. Barabasi. Controllability of Complex Networks, *Nature*, 473, pp. 167-173, 2011.

A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8, pp. 99-118, 1977

R. Manthney and F. Bry. SATCHMO: a theorem prover implemented in Prolog. Proc. of 9th International Conference on Automated Deduction, pp. 415-434. 1988

C. E. Martin and C. K. Riesbeck. Uniform Parsing and Inferencing for Learning. *Proc. of AAAI*, 1986.

C. Matuszek, J. Cabral, M. Witbrock, J. DeOliveira, An Introduction to the Syntax and Content of Cyc, *AAAI Spring Symp. on Formalizing and Compiling Background Knowledge and Its Applications to KR and QA*, CA, March 2006.

C. Matuszek, M. Witbrock, R. Kahlert, J. Cabral, D. Schneider, P Shah, D. B. Lenat, Searching for Common Sense: Populating Cyc from the Web, *Proc. of AAAI*, 2005.

Mezard, M., Parisi, G. and Zecchina, R. (2002). Analytic and Algorithmic Solution of Random Satisfiability Problems. *Science,* Volume 297, pp. 812-815

M. Mitchell. Complex Systems: Network Thinking. *Artificial Intelligence*, 170 (1-2), pp. 1194-1212, 2006.

D. Mitchell, B. Selman and H. Levesque. Hard and easy distributions of SAT problems *Proc. of AAAI*, 1992

D. Molla (2006). Learning of Graph-based Question Answering Rules. Proc. of HLT/NAACL Workshop on Graph Algorithms for Natural Language Processing. 2006

D. Molla and J. L. Vicedo (2007). Question Answering in Restricted Domains: An Overview. *Computational Linguistics*, 33 (1), pp. 41-61

F. Morbini and L. K. Schubert. Evaluation of Epilog: a Reasoner for Episodic Logic. *Commonsense 09*, 2009.

R. Mulkar, J. Hobbs, E. Hovy, H. Chalupsky and C. Lin Learning by Reading: Two Experiments. Third Intl. Workshop on Knowledge and Reasoning for Ans. Questions, 2007

M. Nowak. *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press, 2006.

A. Ntoulas, J Cho and C. Olston. What's New on the Web? The Evolution of the Web from a Search Engine Perspective. *Proc. of WWW*, 2004

B. J. Peterson, Anderson, W. A. and Engel, J. (1998). Knowledge Bus: Generating Application-focused Databases from Large Ontologies. *Proc. of 5th KRDB Workshop,* Seattle, 1998.

J. Prager, J. Chu-Carroll and K. Czuba (2004). Question Answering Using Constraint Satisfaction: QA-by-Dossier-with-Constraints. *Proc. of ACL*, 2004

D. Ramachandran, P. Reagan and K. Goolsbey. First-Orderized ReasearchCyc: Expressivity and Efficiency in a Commonsense Ontology. *AAAI Workshop on Contexts and Ontology: Theory, Practice and Applications*, 2005

D. Ramachandran and E. Amir, Compact Propositionalizations of First Order Theories, *Proc. of AAAI*, 2005

D. Ravichandran and E. Hovy. (2002). Learning Surface Text Patterns for a Question Answering System. *Proc. of ACL,* 2002

B. Richards and R. Mooney (1992). Learning Relations by Pathfinding. *Proc. of AAAI*, 1992

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

S. Russell, and E. Wefald (1991). *Do the Right Thing*. MIT Press.

Schaeffer, J., Bjornsson, Y., Burch, N., Kishimoto, A., Muller, M., Lake, R., Lu, P. and Sutphen, S. (2005). Solving Checkers, *Proc. of IJCAI*, pp. 292-297.

G. Sutcliffe 2009. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0, *Journal of Automated Reasoning*, 43(4), 337-362

B. Selman and H. Kautz. Knowledge Compilation and Theory Approximation, *Journal of ACM*, pages 193-224, 1996.

A. Sharma and K. D. Forbus. Modeling the Evolution of Knowledge and Reasoning in Learning Systems, AAAI Fall Symposium on Commonsense Knowledge, Arlington, VA, 2010

M. E. Stickel. A Nonclausal Connection-graph Resolution Theorem-Proving Program, *Proc. of AAAI*, 1982

Y. Shoham and K. Leyton-Brown. Multi-agent Systems Algorithmic, Game-Theoretic and Logical Foundations. Cambridge University Press, 2009.

P. Tang and F. Lin. Discovering Theorems in Game Theory: Two-Person Games with Unique Pure Nash Equilibriums Payoffs. *Proc. of IJCAI*, 2009

M. E. Taylor, C. Matuszek, P. Smith and M. Witbrock (2007). Guiding Inference with Policy Search Reinforcement Learning. *Proc. of FLAIRS*, 2007

M. E. Taylor, C. Matuszek, B. Klimt and M. Witbrock (2007). Automatic Classification of Knowledge into an Ontology, *Proc. of FLAIRS*, 2007

T. Walsh. Search in small world graphs. *Proc. of IJCAI*, 1999.

T. Walsh. Constraint Patterns. *Proc. of CP*, 2003

Wallace, R. J. and D. Grimes (2008). Experimental Studies of Variable Selection Strategies based on Constraint Weights. *Journal of Algorithms,* 63, pp. 114-129, 2008

D. J. Watts. A Simple Model of Global Cascades on Random Networks. *Proc. of Nat. Acad. of Sciences*, 99(9), 2002.